IOWA STATE UNIVERSITY
Digital Repository

Retrospective Theses and Dissertations

Iowa State University Capstones, Theses and Dissertations

1-1-2006

Semi-automatic transfer function generation for non-domain specific direct volume rendering

Andrew Scott Menz
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Recommended Citation

Menz, Andrew Scott, "Semi-automatic transfer function generation for non-domain specific direct volume rendering" (2006). *Retrospective Theses and Dissertations*. 19342.
https://lib.dr.iastate.edu/rtd/19342

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

Semi-automatic transfer function generation for non-domain specific direct volume rendering

by

Andrew Scott Menz

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Dirk Reiners, Major Professor
Chris Harding
Dimitris Margaritis

Iowa State University

Ames, Iowa

2006

Copyright © Andrew Scott Menz, 2006. All rights reserved.

Graduate College

Iowa State University

This is to certify that the master's thesis of

Andrew Scott Menz

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

Table of Contents

Abstract

The field of volume rendering is focused on the visualization of three-dimensional data sets. Although it is predominantly used in biomedical applications, volume rendering has proven useful in fields such as meteorology, physics, and fluid dynamics as a means of analyzing features of interest in three-dimensional scalar fields. The features visualized by volume rendering differ by application, though most applications focus on providing the user with a model for understanding the physical structure represented in the data such as materials or the boundaries between materials. One form of volume rendering, direct volume rendering (DVR), has proven to be a particularly powerful tool for visualizing material and boundary structures represented in volume data through the use of transfer functions which map each unit of the data to optical properties such as color and opacity. Specifying these transfer functions in a manner that yields an informative rendering is often done manually by trial and error and has become the topic of much research. While automated techniques for transfer function creation do exist, many rely on domain-specific knowledge and produce less informative renderings than those generated by manually constructed transfer functions. This thesis presents a novel extension to a successful semi-automated transfer function technique in an effort to minimize the time and effort required in creation of informative transfer functions. In particular, the method proposed provides a means for the semi-automatic generation of transfer functions which highlight and classify material boundaries in a non-domain specific manner.

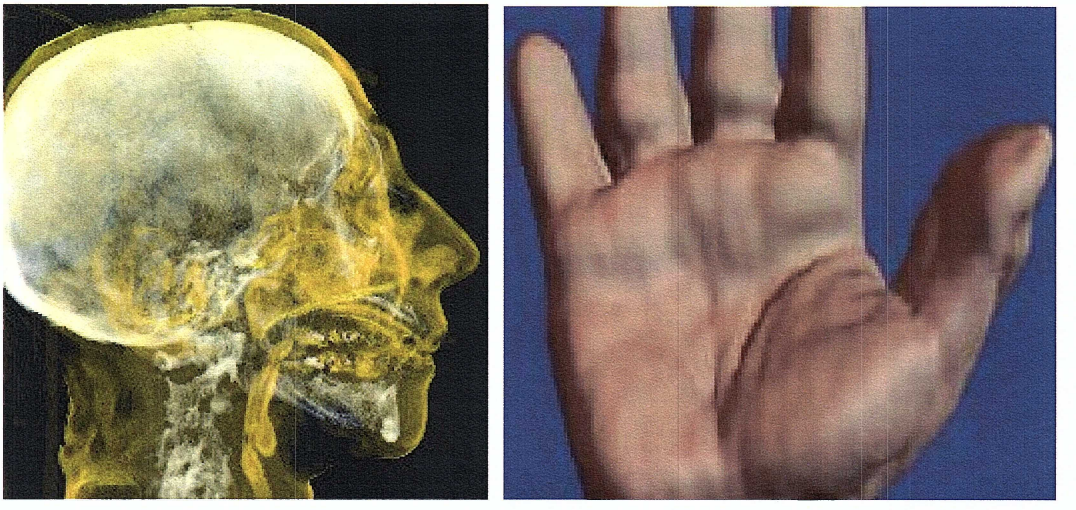
Chapter 1. Introduction to Volume Rendering

1.1 Volume Data

The tools used in many scientific fields for the quantization of three-dimensional objects yield volumetric data sets. These data sets are often stored as scalar fields along a three-dimensional rectilinear grid. Non-rectilinear grids are often resampled to a rectilinear grid to allow efficient processing. Each cubic, face-adjacent cell composed by the grid is called a voxel, and each voxel is assigned a scalar value (or multiple values) over the range 2^n (where n is the number of bits used in the encoding). Medical imaging technologies such as MRI, CT, and PET can yield volumetric data sets in this manner by sampling various material properties at small (mm) intervals and then using reconstruction techniques to assign scalar values to each voxel in a grid of desired dimensions. The values stored in the volumetric data vary depending on the measurement technique used. For example, the voxel values of volumetric data produced using MRI technologies may represent the density of hydrogen nuclei (amongst other properties), whereas data sets generated by CT technologies contain information on the radio-opacity of tissue. In fields such as geology, volumetric data may be generated by measuring seismic energy reflected from geologic layers [SMG03], while the values stored in meteorological volume data sets may represent radar reflectivity of the atmosphere. Additionally, volume data may vary in the number of scalar values assigned to each voxel (multi-dimensional data). Volume data obtained from MRI may contain 3 values per voxel, whereas that obtained from CT typically has only 1 value per voxel. While the material properties quantized in the volume data, and the structure of the data itself, vary according to the measurement methods, the problem of volume visualization is universal.

The primary focus of the field of volume rendering is the accurate and useful visualization of volume data. In biomedical fields, where volume rendering is frequently used, a "useful" visualization is often one that shows regions within the volume corresponding to either materials or material boundaries (the regions between distinct materials). For example, consider a volume data set of a human head obtained from a CT scan. A visualization of material boundaries should show the boundary between soft tissue and bone, air and soft tissue, etc. in an accurate and complete fashion in order to provide an

informative rendering (Figure 1 (a)). While the visualization of materials (as opposed to material boundaries) may also provide informative renderings [Lai95], this thesis focuses primarily on material boundary visualization.



(a) (b)

Figure 1: Comparison of volume rendering techniques. Image (a) shows an example of direct volume rendering of a human head. Image (b) shows an image rendered using indirect volume rendering (from [Lai95] Figure1.1).

1.2 Volume Rendering Techniques

The most common volume rendering paradigms are the indirect and direct method. Indirect volume rendering (IVR) uses surface extraction techniques [LC87] to generate adjacent polygons representing isosurfaces in the volume. That is, the algorithm attempts to identify surfaces, or 3D contours, of similar value in the volume and visualize these surfaces using shaded polygons (Figure 1 (b)). While indirect volume rendering is widely used, the display of isosurfaces based solely on data value can yield confusing or erroneous visualizations in noisy volume data sets. For example, noise in the radio-opacity measurements of a CT scan often results in the assignment of not a single scalar value to a particular material, but a range of values. If multiple materials' ranges overlap in value, detection of boundaries (isosurfaces) becomes a difficult task. In addition, measurement artifacts can influence the shape of the isosurface such that it does not correspond to a material boundary within the volume.

Direct volume rendering (DVR) is a newer approach to volume visualization that has been the focus of much research in recent years, and has been facilitated by advances in graphics hardware including increased texture memory accessibility. The fundamental difference between DVR and the indirect method is that DVR uses every voxel (not just isosurfaces) in the final visualization [Lev88]. DVR uses a *transfer function* which maps each voxel of the volume to optical properties such as color and opacity. These optical properties are then composited in a back-to-front method, and shading, such as the Phong model [Pho75], is applied to produce the final rendering which can then be rotated, sliced, and zoomed to provide interactive exploration of the volume data.

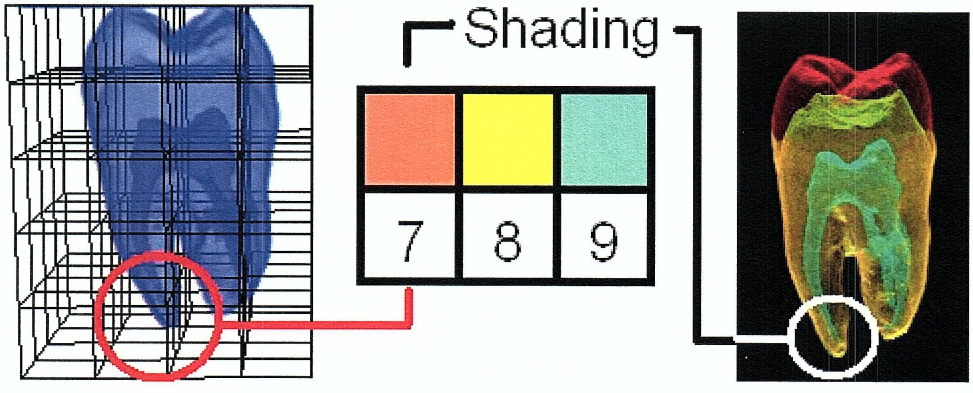
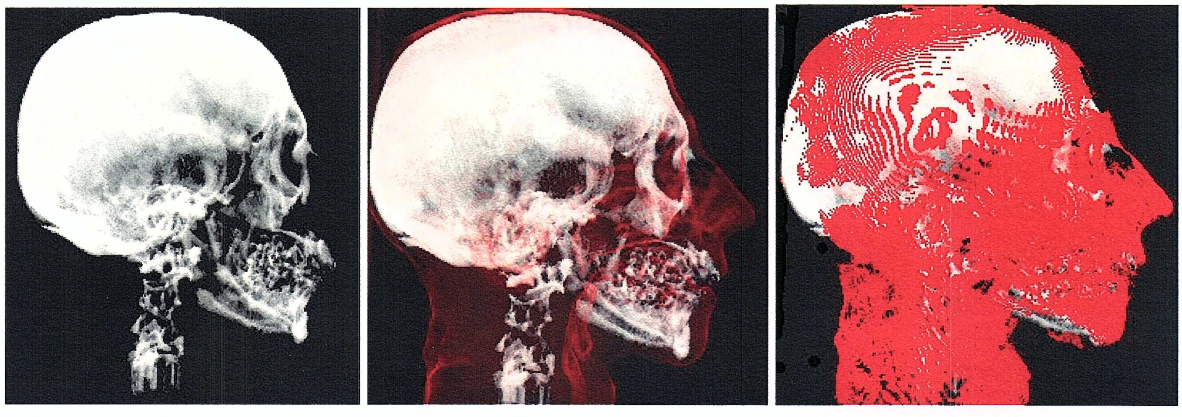


Figure 2 : Each voxel is assigned a data value which corresponds to optical properties in the look up table. After optical properties are determined, shading is applied to produce the final rendered image.

1.3 Transfer Functions

As the critical component of DVR, successful specification of the transfer function has proven to be a notoriously difficult problem. When attempting to visualize material boundaries, each voxel within a volume can be classified as either a material voxel or a boundary voxel (i.e. materials versus where materials meet). The first difficulty in establishing the transfer function mapping then is to determine which voxels in the volume correspond to material boundaries. Ideally only the boundary voxels should be rendered opaque, while materials are assigned lower (or no) opacity (Figure 3 (b)). If the mapping misclassifies too many material voxels as boundary voxels, the final rendering will not accurately convey the location of the material boundaries. Misclassifying too many

boundary voxels as material voxels will result in poorly defined, "spotty" boundaries in the final image. Additionally, mapping any boundary to an opacity that is too high may occlude boundaries within the volume, that is, one boundary may hide another boundary underneath (Figure 3 (c)). Thus care must be taken to map each voxel to an appropriate opacity.



(a) (b) (c)

Figure 3: Direct volume renderings of the human head. In (a) the material boundary between bone and soft tissue (white) is clearly rendered. The boundary between soft tissue and air is shown in red in (b) without occluding the boundary behind it. In (c) the air/soft tissue boundary is rendered spotty and too opaque, resulting in occlusion and an uninformative rendering.

The second major obstacle in transfer function specification is *segmentation*, that is, determining which boundary voxel belongs to which distinct material boundary within the volume. While material/boundary voxel classification is generally indicated using opacity, modification of voxel color is a useful method for visually indicating boundary classification. For example, consider again a volume data set of a human head. Within the volume data are voxels which compose the boundary between soft tissue and bone (boundary X), as well as voxels which compose the boundary between air and soft tissue (boundary Y). An informative transfer function should then be able to map voxels from boundary X to some color, say white, and map voxels from boundary B to another distinguishable color, say red, in the final rendering such that the user can immediately differentiate the two boundaries (Figure 3 (b)).

Although other properties of transfer functions may be useful in various applications, this thesis uses the following general criteria for qualitative analysis of transfer functions.

When attempting to visualize material boundaries, a transfer function is successful if it provides an informative mapping such that:

1. Voxels within material boundaries are mapped to higher opacity values than non-boundary voxels.
2. No material boundary in the final rendering is occluded by any other boundary or material.
3. Each material boundary is consistent and relatively uninterrupted.
4. Each material boundary is visually distinguishable from every other.

Early implementations of DVR used one-dimensional transfer functions for boundary visualization. These transfer functions map each voxel to an opacity based solely on the data value of the voxel. The advantage of these rudimentary transfer functions is simplicity of specification and calculation. Formulation of the mapping is as simple as assigning an opacity (and perhaps color) to each possible data value. This method proves useful in data sets with known material value ranges and minimal noise, such as particularly clean biomedical volume data sets. Organic materials tend to have relatively small value ranges within CT and MRI data sets, which facilitates creation of one-dimensional transfer functions of this kind. For example, if human bone is known through experimentation to have MRI density values in the range [W, Y], then a transfer function can be generated that assigns higher opacity at values W and Y – thus making the boundary of the material more opaque. Unfortunately, in practice biomedical data sets assign a wide range of values to each material, and the value ranges of multiple materials frequently overlap. So, if in the previous example there exists another material B with known density value range [X, Z] and $W < X < Y < Z$, then a one-dimensional transfer function that assigns high opacity to voxels with value Y will yield a final rendering where some material voxels belonging to material B are assigned high opacity. If the goal of the transfer function is to make only *boundary* voxels opaque, the result is an uninformative image.

The next stage in transfer function evolution took the form of multi-dimensional mappings. Multi-dimensional transfer functions (MDTF's) map from multiple voxel properties (rather than just value) to optical properties such as color and opacity. These voxel properties, which are calculated from voxel values prior to rendering, may include gradient, gradient magnitude, second directional derivative measures, curvature measures, and others. By using additional voxel properties, multi-dimensional transfer functions allow

a greater degree of freedom in mapping specification, and thus increase the expressiveness of the transfer function. The expressiveness of MDTF's can help overcome the shortcomings of one-dimensional transfer functions by providing additional measures on which to assign optical properties to voxels. As an example, consider voxel A_1 with data value V_1, and voxel A_2 with data value V_2. Assume voxel A_1 is within material M_1 and voxel A_2 is within material M_2 ($A_1 \neq A_2$, $M_1 \neq M_2$). A one-dimensional transfer function which maps voxel value to optical properties is not able to distinguish voxel A_1 from voxel A_2 as they have the same value. A MDTF, on the other hand, may determine that voxels A_1 and A_2 differ in some derived property (e.g. gradient magnitude), and is then capable of assigning different optical properties to each voxel.

One particular breed of MDTF's that has proven particularly effective in material boundary visualization makes use of data value and two derived voxel properties – gradient magnitude, and a second directional derivative measure. This class of transfer functions (and their automated generation) was first proposed by Kindlmann (reference) and forms the foundation of the research presented in this thesis. As such, a detailed explanation of the theory, implementation, and automated generation of these functions is presented in Chapter 4.

1.4 Transfer Function Generation

As mentioned previously, two major difficulties in transfer function specification are classifying materials versus voxels, and material boundary segmentation. Utilities designed for manual transfer function creation provide varied means for overcoming these difficulties, and have proven useful in providing informative renderings. Unfortunately, manual transfer function specification is often time consuming and requires a trial-and-error methodology to produce an informative rendering. Additionally, these utilities are often complex and require significant application specific expertise to produce useful transfer functions.

A good deal of research has been invested in simplifying transfer function specification through semi-automated generation techniques. Automated segmentation techniques can successfully classify materials to a high degree of accuracy, but often require very specific conditions in order to be successful. For example, segmentation algorithms

based on machine learning principles require a significant number of classified volume data sets for training. Other segmentation methods are based on domain-specific knowledge, such as commonly measured values for expected materials. Still others focus only on data sets from one measurement modality, such as MRI [Lai95], or require the user to manually specify small regions of each material [ISNC03]. In his thesis, Kindlmann proposes a means for semi-automated transfer function generation that has proven effective at classifying materials versus boundaries, but lacks boundary segmentation. The goal of this thesis is to extend Kindlmann's method through the addition of a novel means of segmentation of non-domain specific volume data. The addition of segmentation to Kindlmann's method not only allows for boundary classification via color selection, but also allows for greater accuracy in specifying appropriate opacity to known material boundaries resulting in greater user modification capabilities and a more informative rendering.

Chapter 2. Previous Work

The use of multi-dimensional transfer functions for DVR was pioneered by Levoy [Lev88] who assigns opacity to voxels based on gradient magnitude (f') and data value such that voxels with larger gradient magnitude are assigned higher opacity. The intuition behind this procedure is the notion that boundary voxels (those that correspond to a boundary between material regions) should exist in a region of the volume where data values change relatively rapidly and material voxels (those corresponding to one material) should exist in a region of the volume with a relatively constant data value. By this method of opacity assignment, voxels corresponding to boundaries between materials are rendered more opaque, whereas voxels corresponding to materials are assigned little or no opacity. The rendered image resulting from this technique then primarily displays the location of material boundaries within the volume. The development of multi-dimensional transfer functions which focus on displaying material boundaries was furthered by a good deal of later research. Lum [LM04] extends Levoy's research by enhancing surface visibility through the modification of surface shading. This allows variance in opacities to represent the thickness of materials in the volume, resulting in a more informative rendering. More generally, both [PLSea00] and [MHBea00] provide a fairly comprehensive overview of the various techniques used in boundary visualization through DVR.

In [KKH02] Kniss proposes a means for specifying these MDTF manually through an interactive, data-centric technique. This manual specification makes use of a three-dimensional transfer function consisting of data value, first, and second directional derivative measures (the same type of transfer functions used in this thesis), and allows the user to manually specify the transfer function in a variety of ways to produce renderings in which higher opacity is assigned to boundary voxels and different material boundaries are rendered in different colors. One means by which these transfer functions can be specified is by directly creating a mapping from data value and derivative measures to optical properties. Another, more novel, means of MDTF specification proposed by Kniss allows the user to select regions of the *volume* space which are then mapped to data value and derivative measures, which in turn can be mapped to optical properties.

In an effort to minimize the difficulty and tedium of manual specification, several methods exist for the semi-automatic specification of MDTF's. He et al. develop MDTF's through a stochastic search of parameter space using supervised and unsupervised genetic algorithms [HHKea96]. The supervised algorithm relies on the user to select from a collection of thumbnail renderings, whereas the unsupervised version relies on a user-specified fitness function to determine the optimal rendering parameters.

Other efforts attempt to improve the rendering of boundaries through automated segmentation (material classification). Laidlaw introduces a semi-automatic method of segmentation for volume data obtained from MRI measurement devices [Lai95]. Volume data generated from MRI is sensitive to at least three data-collection parameters such that modifying these parameters appropriately helps delineate the data values of each material in the volume. Laidlaw first uses a goal-based data-collection technique to optimize these parameters, and then uses a novel tissue classification technique on the volume data produced to classify each material. More recently, Kniss explores a novel means of visualizing multiple segmentations produced by statistical segmentation algorithms [KUAea05]. An overview of the application and effectiveness of various machine learning techniques for volume segmentation is discussed in [CLOea05].

Chapter 3. Material Classification

The direct volume rendering method presented in this thesis is based on established methods of semi-automatic transfer function generation and volume segmentation. As such, section 3.1 below details the existing method of segmentation upon which the proposed method is based, and makes reference to its limitations. Section 3.2 details the proposed segmentation method and its advantages. The extension of Kindlmann's transfer function generation method by way of this segmentation is explained in Chapter 4.

3.1 Confidence Connected Region Growing

Recall that the goal of segmentation within a volume data set is the classification of individual materials. While many segmentation schemes exist, the method proposed in this thesis attempts to reduce classification dependence on the user's domain knowledge by minimizing user input to four scalar parameters. The method proposed is based on a form of *region growing* commonly used in computer imaging, termed the "confidence connected" method by Ibanez [ISNC03]. Confidence connected region growing begins by specifying one *seed* for each material to be classified. The seed consists of multiple voxels that belong to one and only one material in the volume. The mean and standard deviation of each region's voxel values are calculated. Each seed is then *grown* within the volume according to a confidence parameter, that is, every non-region voxel adjacent to the seed's region is included in the region if it falls within a specified range of the region's mean. In this way, each seed (representing a material) grows within the volume to include all neighboring voxels that are likely to belong to this same material. Seed are grown iteratively until every voxel in the volume has been assigned a material (or a specified number of iterations are processed).

One of the strengths of region growing as a segmentation scheme is that it works well on volume data with limited data value contrast between materials. However, region growing requires manual specification of seeds to get the algorithm started. While this may not prove problematic for expert users, those who are unfamiliar with the data's domain or working in a new domain may find it difficult to locate each material and accurately specify a

seed. Additionally, while data acquired in, say, biomedical fields may have a known structure, data acquired in other fields such as geology may not have a known structure. For example, when looking at an unsegmented rendering of a human head, a medical professional may find it a simple task to identify within the rendering a small set of voxels representing bone based on previous experience with similar volume renderings. On the other hand, a geologist viewing an unsegmented rendering of geological layers may not know the structure of materials in the volume a priori, and thus may find it difficult to specify seeds within the volume. *In an effort to minimize user input and simplify the segmentation process, the segmentation method proposed here simplifies segmentation of single-valued volume data to the selection of four scalar parameters.*

3.2 Proposed Segmentation Method

The following section explains the segmentation algorithm proposed in this thesis. Terms and functions are first defined, proceeded by the algorithm itself. Each step of the algorithm is then explained and justified.

Let O be a real-world object, and D be the volume data set generated by measuring material properties of O. Let V be a voxel in D. Define the function Val(V) as the data value of voxel V, Corr(V) as the material in O to which voxel V corresponds, and GMag(V) as the gradient magnitude of the data value of voxel V. The term *material* is used to denote a homogeneous spatial region of a real-world object. A *material region* within D is defined as a set of voxels R, such that for all voxels V in R, Corr(V) = M for some material M in O. Additionally, a voxel is said to be in the *interior* of a material region if each of the voxel's 26 adjacent neighbor voxels is in the same material region. Further, let M be a material region with mean data value M_{avg} and standard deviation σ. The *adjacency merge range* $\text{Range}_{adj}(M)$ is the data value range $[M_{avg} - J_A * \sigma,\ M_{avg} + J_A * \sigma]$ for scalar parameter J_A, and the *global merge range* $\text{Range}_{global}(M)$ is the data value range $[M_{avg} - J_G * \sigma,\ M_{avg} + J_G * \sigma]$ for scalar parameter J_G.

Segmentation Algorithm

1 Set gradient magnitude threshold parameter GM_{mat}
2 For each voxel V in the volume:
3 If GMag(V) < GM_{mat}
4 Mark V as an interior voxel

5 Assign all adjacent interior voxels to the same material region

6 Set gradient magnitude threshold parameter GM_{bound}
7 For each voxel V in the volume:
8 If GMag(V) > GM_{bound}
9 Mark V as a boundary voxel

10 Set adjacency joining parameter J_A
11 While some material region R is still growing
12 Set $R' = R$
13 For each voxel V in R'
14 For every neighboring voxel Q of V
15 If Val(Q) is in $\text{Range}_{adj}(R)$ and Q is not a boundary voxel
16 Include Q in R
17 Recalculate $\text{Range}_{adj}(R)$

18 For each material region R_1 which is adjacent to material region R_2
19 if $\text{Range}_{adj}(R_1)$ intersects $\text{Range}_{adj}(R_2)$
20 Merge R_1 and R_2 into a single material region R_3
21 Calculate $\text{Range}_{adj}(R_3)$

22 Set global joining parameter J_G
23 For each material region R
24 Calculate $\text{Range}_{global}(R)$
25 For each material region R_1
26 For each material region R_2 s.t. $R_1 \neq R_2$
27 If $\text{Range}_{global}(R_1)$ intersects $\text{Range}_{global}(R_2)$
28 Merge R_1 and R_2 into a single material region R_3
29 Calculate $\text{Range}_{global}(R_3)$

The first step in the proposed segmentation method is to determine the seeds needed for the region growing algorithm. Recall that seeds are small areas of each material in the volume, and are generally specified by hand. Note that by the definition of material region above, the process of specifying seeds can be restated as the process of specifying material regions since each material region corresponds to exactly one material in the real world object. Rather than specifying these regions manually, the algorithm above uses four related

parameters GM_{mat}, GM_{bound}, J_A, and J_G to specify seeds semi-automatically and grow each seed within the volume.

The GM_{mat} parameter is a scalar threshold on the gradient magnitude of data value. In the loop at line 2, it is used to determine which voxels in the volume are likely to belong to the interior of a material region. Intuitively, a material region is a subset of all voxels in the volume which correspond to only one material in the real-world object. The interior of a material region is then a set of voxels which all correspond to the same material *and* have no adjacent neighbors which correspond to any other material. It is this property of interior voxels that makes them ideal for seeds in the volume. Once these interior voxels are identified, the algorithm can then form sets of adjacent interior voxels, and assume with high confidence that these sets represent material regions (that is, no two voxels in any set correspond to different materials).

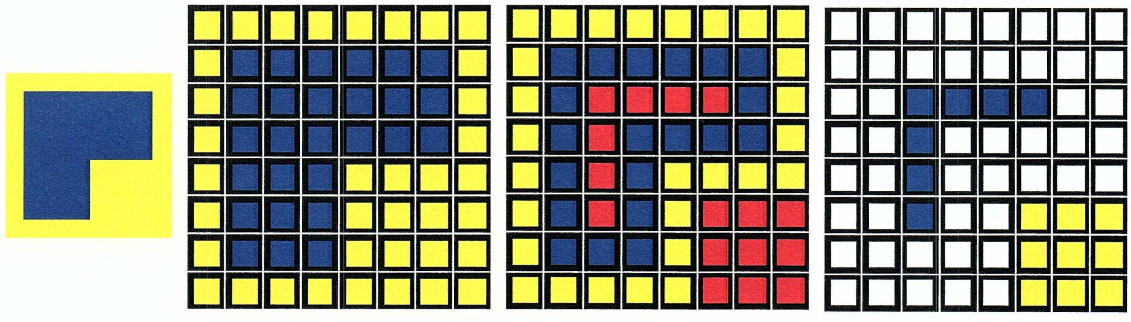
The following logic illustrates the motivation behind this method in the ideal case.

Temporary Assumption:
Val(V) = Val(W) if and only if Corr(V) = Corr(W).

Claim 1: GMag$(V) = 0 \Rightarrow V$ is in the interior of a material region R.
Proof: (by contradiction)
Assume V is not in the interior of R and GMag$(V) = 0$.
V not in the interior of R $\Rightarrow \exists$ voxel X s.t. Corr$(X) \neq$ Corr(V)
$\Rightarrow$ Val$(X) \neq$ Val(V) (by Temporary Assumption)
$\Rightarrow$ GMag$(V) > 0$
$\Rightarrow \Leftarrow$.
Thus V is in the interior of R.

Let V_1, V_2 be voxels in the interior of material regions R_1, R_2 s.t. $V_1 \in R_1$, and $V_2 \in R_2$.
Claim 2: $R_1 \neq R_2 \Rightarrow V_1$ and V_2 are not adjacent.
Proof: (by contradiction)
Assume V_1 and V_2 are adjacent and $R_1 \neq R_2$.
V_1 an interior voxel $\wedge$ V_1, V_2 adjacent $\Rightarrow$ Corr(V_1) = Corr(V_2)
$\Rightarrow R_1 = R_2$ (by definition of material region)
$\Rightarrow \Leftarrow$.

The temporary assumption above states that every distinct material in a real-world object has exactly one corresponding value in the volume data. Under this assumption, Claim 1 shows that any voxel with zero gradient magnitude is within the interior of one of the material regions in the volume. Claim 2 shows that no two interior voxels can be adjacent unless they correspond to the same material. Thus, if the temporary assumption holds true, every set of adjacent voxels with zero gradient magnitude is a material region, and is therefore a seed for region growing. To illustrate this point pictorially in two dimensions, Figure 4 (a) shows a two-dimensional object consisting of two materials, yellow and blue. Figure 4 (b) then shows the object represented by a grid of pixels (similar to a slice of a volume data set), where a blue pixel indicates the pixel corresponds to the blue material, and all blue pixels form a material region (similarly for yellow pixels). If one assumes that blue pixels all have the same data value B, and yellow pixels all have the same data value Y (where $B \neq Y$), then the red pixels in Figure 4 (c) can be interpreted in two ways. First, red pixels can represent every pixel that has zero gradient magnitude, that is, the difference between the data value of each red pixel and its eight neighbors is zero (Claim 1). Secondly, the red pixels can represent every voxel that has no neighbor which does not correspond to the same material as that pixel (interior pixels). Then creating sets from all adjacent red pixels produces two distinct material regions (Claim 2) in Figure 4 (d) such that each material region may serve as a seed for region growing. This example then naturally extends into a three dimensional object represented with voxels.



(a) (b) (c) (d)

Figure 4: Two-dimensional example of sets of adjacent interior voxels.

Unfortunately, the temporary assumption does not hold true in practice. Due to noise in the volume data and band-limiting during data acquisition (discussed further in section 4.1), material properties measured for a specific material are rarely constant. In fact, every effective segmentation scheme must assume that each material has a *range* of data values in the volume data, thus voxels in the interior of a material region may have non-zero gradient magnitude. Because of this, using a zero gradient magnitude criteria for determining interior voxels may produce a very small set of voxels for each material region, and in the worst case may produce an empty material region for some material. To account for variations in data value amongst voxels corresponding to the same material, the gradient magnitude constraint for determining interior voxels is relaxed to a small positive threshold GM_{mat} (line 1).

Specifying material regions in the method described in lines 1 through 5 introduces complications not found in manual seed specification. If the GM_{mat} threshold is set too high the use of adjacency as a material region set inclusion criteria (line 5) may allow "bleeding" between two distinct material regions, effectively joining multiple sets of interior voxels which correspond to different materials. That is, if there is a high gradient magnitude voxel path from one material region to another, these may be erroneously merged together. If GM_{mat} is set too low, the sets of adjacent interior voxels may not be large enough to act as seeds for the region growing that occurs later in the algorithm. Thus, the successful selection of material regions (seeds) is highly dependent on the specification of an appropriate gradient magnitude threshold.

The success of the region growing section of the algorithm (lines 10 -17) is partially dependent on another scalar parameter GM_{bound} which is a gradient magnitude threshold similar to GM_{mat}. The algorithm uses GM_{bound} (lines 6 – 9) to classify all voxels greater than the threshold as boundary voxels. As discussed in section 4.1, voxels with high gradient magnitude generally correspond to a boundary between materials in the real-world object. Boundary voxels serve as a natural spatial division between distinct materials. This property is exploited in the algorithm (line 15) to limit the growth of each material region, which reduces the possibility of two material regions (which correspond to different materials) from becoming adjacent during region growing.

The region growing section also relies on the adjacency joining parameter J_A which determines a data value range $\text{Range}_{adj}(R)$ for each material region R based on the region's mean and standard deviation. Each voxel of each region material R is grown to include all adjacent non-boundary voxels which have a data value within $\text{Range}_{adj}(R)$. In this way, each material region is grown to include only voxels which correspond to the region's material with high confidence. After each voxel in a region R is grown, $\text{Range}_{adj}(R)$ is recalculated based on the new mean and standard deviation.

Lines 18 – 21 of the algorithm address an additional complication that arises from using gradient magnitude to specify seeds in the volume. Noise in the volume data may yield areas of voxels with large gradient magnitude that fall above the GM_{mat} threshold and thus are not considered for inclusion in a material region. As a result, before region growing (after line 5) the algorithm may generate multiple non-adjacent material regions which correspond to the same material in the real-world object. In order to correctly segment the volume after region growing it is necessary to identify which of these material regions should be merged into a single material region. For each material region R, the algorithm determines which other material regions R has become adjacent to as a result of region growing. If any of these regions Q has $\text{Range}_{adj}(Q)$ which intersects $\text{Range}_{adj}(R)$, Q and R are merged into the same material region. Similar to the logic behind voxel inclusion in a material region during region growing (line 14), the algorithm attempts to merge material regions that are believed with high confidence to correspond to the same material (line 19).

To understand the motivation behind this logic, consider a volume data set that contains voxels representing material M and two material regions R_1 and R_2 formed by grouping adjacent voxels with gradient magnitude below GM_{mat}, where $\text{Corr}(R_1) = \text{Corr}(R_2) = M$, and $R_1 \cap R_2 = \emptyset$. If one assumes that each material is relatively homogeneous, and that the material properties measured in the volume data are sufficient to differentiate each material, then M should have a unique probability distribution of measured data value. Informally, if each material is different and that difference is detectable in the volume data, then each material should have an average data value that differs from every other material. Moreover, since R_1 and R_2 are subsets of M, R_1 and R_2 should have similar probability distributions (Figure 5 (a)). Then, if two material regions become adjacent after region

growing and have mean values within a small range of each other, the algorithm assumes with high confidence that these two material regions correspond to the same material.

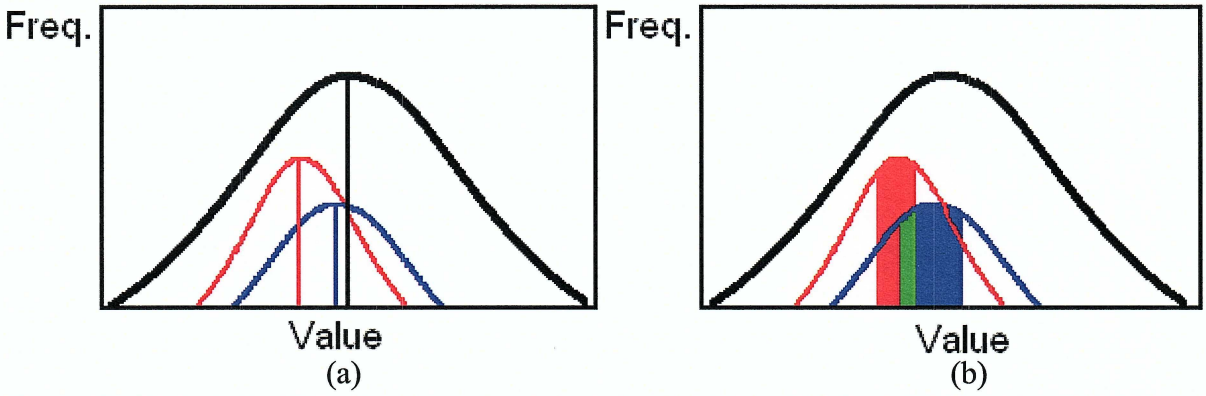


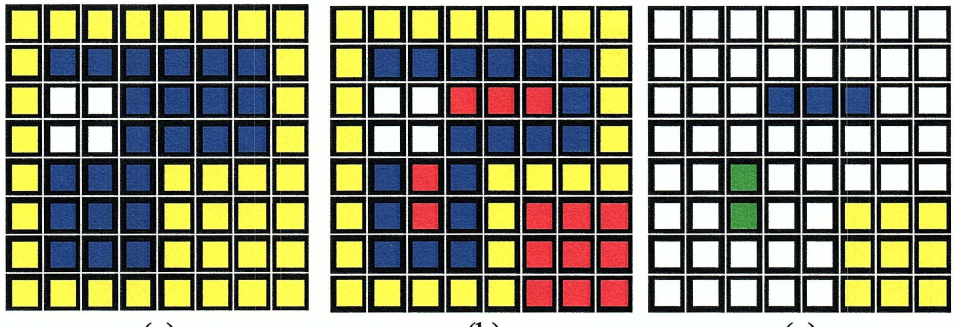
Figure 5: Probability Distributions. Image (a) shows the probability distribution of data value for all voxels corresponding to material region M (black), R_1 (red), and R_2 (blue). Mean values are indicated with vertical lines. Similarly, in (b) $\text{Range}_{\text{adj}}(R_1)$ and $\text{Range}_{\text{adj}}(R_2)$ are indicated by solid red and blue regions, respectively, and the values where the ranges intersect are colored green.

The final stage in the segmentation algorithm (lines 22 – 29) attempts to merge non-adjacent material regions which correspond to the same material. Consider a volume data set representing material properties of a human body. If each bone in the body is considered to be composed of the same material, then the bones will be represented as non-adjacent material regions in the algorithm before line 22. To accurately classify each of these regions as corresponding to the same material, each of these material regions must be merged into a single material region. The is accomplished using the global merge parameter J_G which is then used to calculate the global merge range $\text{Range}_{\text{global}}(R)$ for each material region R. For each material region R, if any other material region Q has global merge range $\text{Range}_{\text{global}}(R)$ which intersects $\text{Range}_{\text{global}}(R)$, then Q and R are merged into a single material region.

The motivation behind the use of two separate merge parameters J_A and J_G, rather than a single merge parameter, is that material regions need to be merged in two separate instances and with varying levels of confidence. The two instances that necessitate these material region merges are: 1) noise in the data produces a high gradient magnitude "gap" within the interior of a single material region, causing some interior voxels in the material

region to become non-adjacent, and 2) multiple spatially non-adjacent areas of the real-world object are composed of the same material.

In the first instance, the interior voxels of some material region become non-adjacent due to misclassification caused by noise. Recall the two-dimensional object from Figure 4 (a). Now assume that noise is generated during data acquisition which effects the data values of four pixels shown in white in Figure 6 (a). When lines 1 – 4 of the segmentation algorithm are executed, assume the noise causes the gradient magnitude of these pixels to exceed GM_{mat} and as a result these voxels are not marked as interior pixels (Figure 6 (b)). Then when line 5 of the algorithm forms sets of adjacent interior pixels, the interior of the blue material region from Figure 6 (a) is split into two material regions (indicated by blue and green in Figure 6 (c)). When these two regions are grown they will eventually become adjacent and must then be merged. In practice, the adjacency merge parameters J_A can be set fairly high (specifying a relatively high confidence that material regions which grow



(a) (b) (c)

Figure 6: Two-dimensional example of sets of adjacent interior voxels in the presence of noise.

together should be merged) without causing erroneous merging. This is due to the use of the GM_{bound} parameter to classify boundary voxels within the volume. While it is assumed that material voxels correspond to only one material, voxels near the boundary between two materials are assumed to correspond to multiple materials. Note that the region growing section of the algorithm (lines 10-17) does not allow growth into boundary voxels. Since boundary voxels serve as natural separators between materials, stopping material regions from growing beyond boundaries decreases the likelihood that two material regions

corresponding to different materials will become adjacent during region growing. Figure 7 illustrates the the material boundaries in a section of a human tooth (in white).

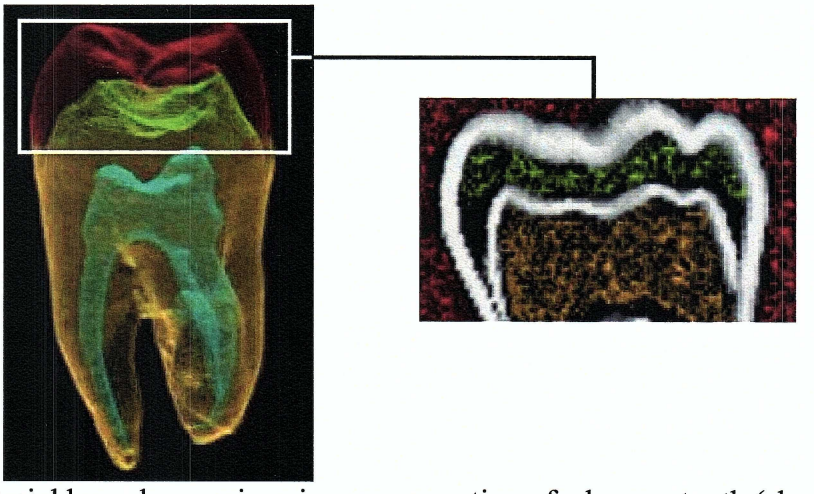


Figure 7: Material boundary regions in a cross-section of a human tooth (shown white).

The second instance that requires merging of material regions occurs when the real-world object measured in the volume data contains areas of one material that are not adjacent. For example, bones in the human body are all composed of the same material, but are not necessarily adjacent. Merging of these material regions is then done in lines 22 – 28 of the algorithm based on a global merge range determined by the scalar parameter J_G. Note, however, that this section of the algorithm merges *any* two material regions with intersecting global merge ranges. Then by setting J_G smaller than J_A the user can specify *how much* more likely a merge is to occur between two adjacent material regions than two non-adjacent material regions.

After completion of region growing, it is assumed that *most* material voxels are classified according to the material to which the voxel corresponds. This segmentation is then used later in the proposed method for two distinct purposes: accurately assigning opacity to boundary voxels through the use of boundary specific histogram volumes (described in section 4.2), and rendering each distinct boundary a different color (as discussed in section 4.7). The method proposed for semi-automatic assignment of opacity to each voxel that makes use of this segmentation is based on a method developed by Kindlmann and is explored in Chapter 4.

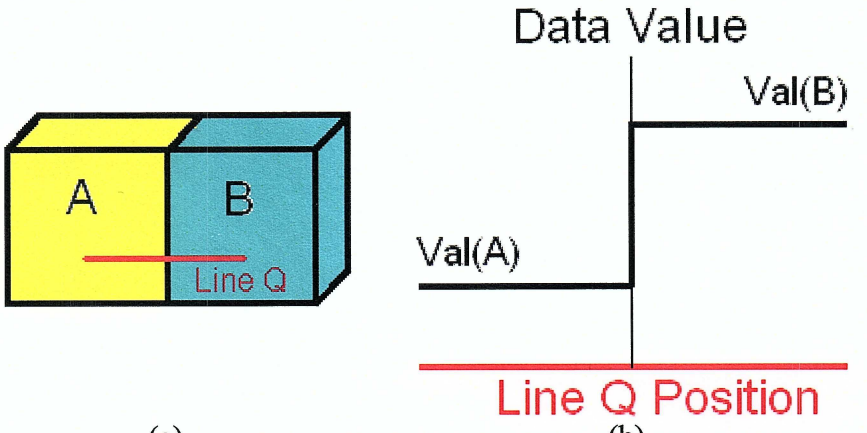
Chapter 4. Semi-Automatic Transfer Function Generation

Kindlmann in his M.S. thesis details a method of semi-automatic transfer function generation for visualizing material boundaries. The transfer function generation method presented in this thesis is based on Kindlmann's method, though it extends Kindlmann's method to take advantage of the segmentation method proposed in section 3.2. The following section contains a brief overview of the theory and justification behind Kindlmann's method, and simultaneously proposes extensions to Kindlmann's method. Note that only topics relevant to this thesis are detailed below – for more detail on the finer notes of Kindlmann's research the reader is encouraged to refer to [Kin99].

4.1 Kindlmann's Boundary Model

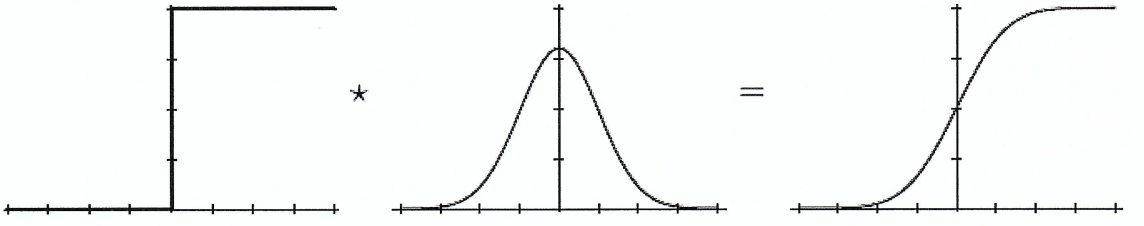
As mentioned previously, the final goal of the transfer functions explored in this thesis is to produce a mapping from voxel to optical properties such that the final rendered image clearly shows all material boundaries in the data set. As such, it's important to understand the properties of boundary voxels that make them distinct from material voxels. To this end, Kindlmann proposes a boundary model representing how data values change within boundary regions. To demonstrate this model consider two physically adjacent materials A and B, where A has value Val(A) and B has value Val(B). If a line segment Q is drawn in 3D space such that one end point of Q is in A, the other is in B, and the middle point of Q is at the boundary between A and B (Figure 8 (a)), one can then graph the data value at each point in the line segment (Figure 8 (b)). Note that in Figure 8 (b) there is a discontinuity in data value at the point in the center of the line segment where A and B meet.

Ideally, volume data obtained from measurement tools would show this same discontinuity in data value at material boundaries. In practice, however, data values are band-limited prior to sampling, resulting in a continuous change of values over position. That is, the data values of some voxels correspond to multiple materials. To model this behavior, Kindlmann assumes that data values at material boundaries are the result of applying a Guassian filter to blur the values (Figure 9). The resulting measurement curve is then the integral of the Gaussian kernel, called the error function, or erf( ).



(a) (b)

Figure 8: Data value transition at a material boundary within an ideal volume.

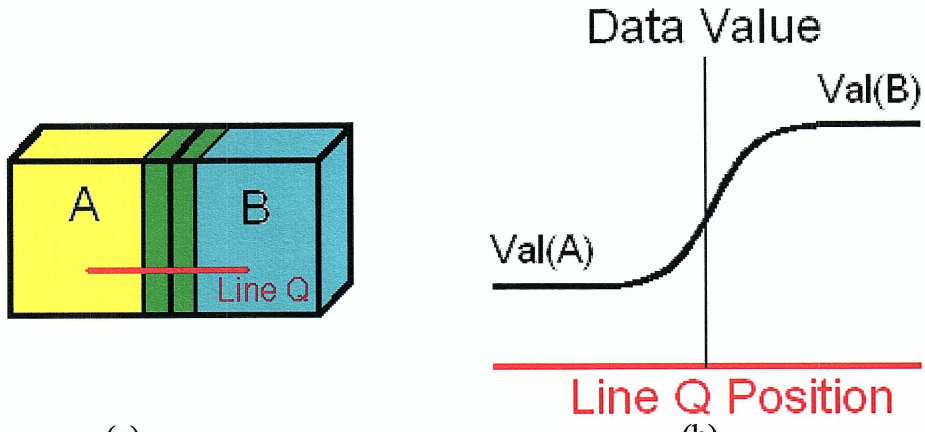


(a) (b) (c)

Figure 9: Result (c) of applying a Gaussian filter (b) to boundary value measurements (a) (from [Kin99] Figure 3.1).

Returning to the previous example, the true measurement of the data values in materials *A* and *B* includes a boundary region where data values transition from Val(*A*) to Val(*B*) gradually (Figure 10 (a)). One can then graph the expected measured data value at each point in Line Q (Figure 10 (b)) to see the expected gradual transition between the two data values.

Graphing the first and second derivative of the expected data value versus line position (Figure 11) show that the derivatives of the data value have distinct properties at the center of the material boundary (represented by a black vertical line in Figure 11). Note that at the middle point of line Q (which lies on the center of the boundary between material A and B) the first derivative of data value reaches its maximum, and the second derivative reaches zero. Considering the middle point of line Q to have *position* zero, it can then be said that any voxel in the the boundary region between materials *A* and *B* is in the center of



(a) (b)

Figure 10: Expected data value transition at a material boundary within a material boundary region. Image (a) shows the material boundary region between *A* and *B* in green. Image (b) graphs the expected gradual transition in data value from Val(*A*) to Val(*B*).

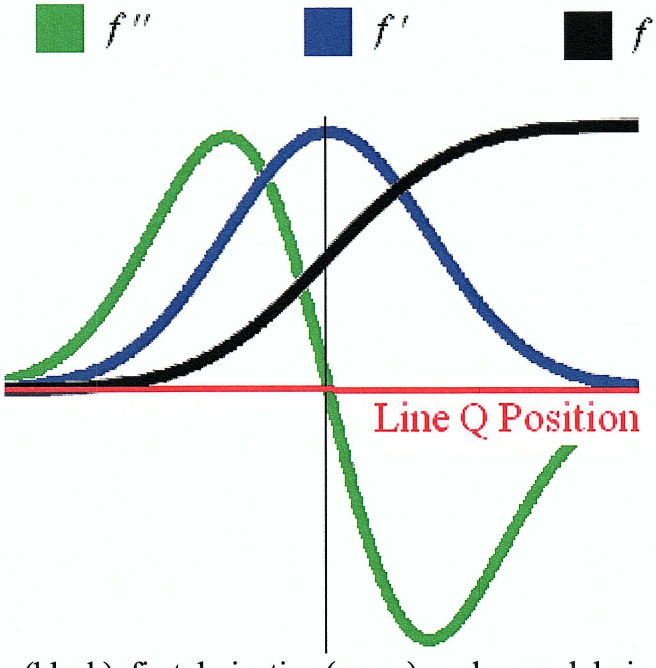


Figure 11: Data value (black), first derivative (green), and second derivative (blue) versus position in line segment Q (modified from [Kin99] Figure 3.4). The black vertical line represents the position in the center of the material boundary, i.e. position zero.

the material boundary if a) the voxel's data value is exactly between Val(*A*) and Val(*B*), b) the voxel's first derivative with respect to position is a maximum, and c) the voxel's second derivative with respect to position is zero. Moreover, any voxel along line Q with position *p* is *p* voxel units away from the center of the material boundary. As a partial goal of the

transfer functions considered in this thesis is the informative rendering of boundary voxels, this position measure can then be used to make voxels closer to the center of the material boundary more opaque, and make voxels further from the center of the boundary less opaque.

More formally, consider a volume with data values in scalar field *f*. Let the *position* of a voxel *V* in a material boundary be roughly defined as the shortest distance (in voxel units) from *V* to any point in the material boundary. Kindlmann's boundary model defines the value *v* of a voxel at position *x* within a boundary between two material with data values v_{min} and v_{max} to be

$$v = f(x) = v_{min} + (v_{max} - v_{min}) \frac{1 + \mathrm{erf}(\frac{x}{\sigma\sqrt{2}})}{2} \quad \text{(Equation 1)}$$

where erf is the error function

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\Pi}} \int_0^x e^{-t^2}\, dt \quad \text{(Equation 2)}$$

The use of scalar parameter σ in Equation 1 roughly determines the "thickness" of the boundary. That is, σ determines how quickly (per voxel unit) values in the boundary transition from v_{min} to v_{max}. Using this boundary model, the first and second derivatives of data value for a voxel in the boundary at position *x* become

$$f'(x) = \frac{v_{max} - v_{min}}{\sigma\sqrt{2\Pi}} e^{-\frac{x^2}{2\sigma^2}} \quad \text{(Equation 3)}$$

$$f''(x) = -\frac{x(v_{max} - v_{min})}{\sigma^3\sqrt{2\Pi}} e^{-\frac{x^2}{2\sigma^2}} \quad \text{(Equation 4)}$$

Figure 12 below shows the relationship between data value, first and second derivative, and σ with respect to position in a boundary. Note the similarity of Figures 11 and 12.

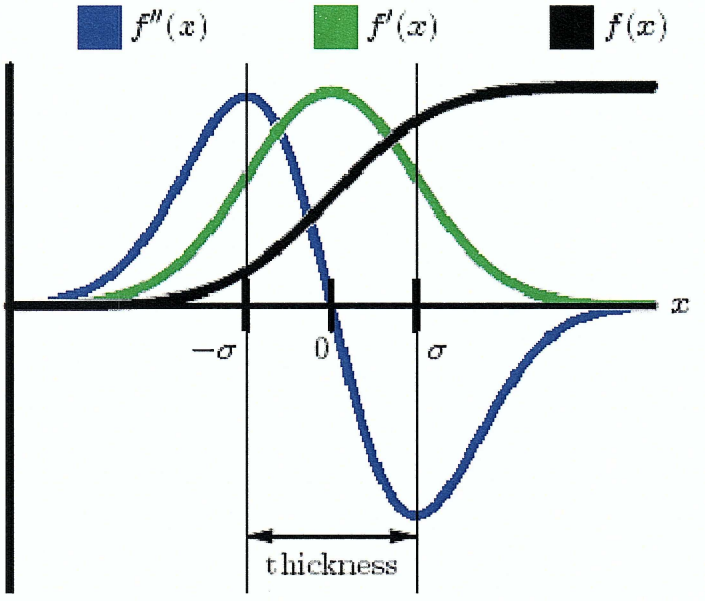


Figure 12: Data value, first, and second derivative versus Kindlmann's position measure (modified from [Kin99] Figure 5.2).

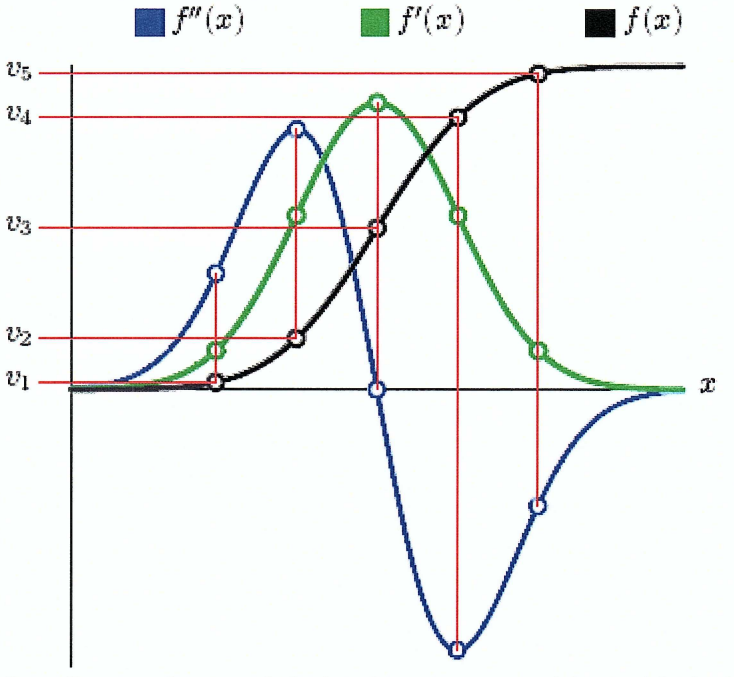


Figure 13: Data value, first, and second derivative versus position with derivative measures for data values v_1 through v_5 plotted (modified from [Kin99] Figure 3.5 (a)).

The goal of Kindlmann's method is to create a mapping from data value to position such that every voxel can be assigned an opacity based solely on data value. Consider then data values v_1 through v_5 within the material boundary (shown in Figure 13). Note that data values within an ideal material boundary increase monotonically, and thus each data value has a unique position, as well as a unique combination of first and second derivative values. This implies there exists a one-to-one relationship between data value and position *within a boundary*. Further, first and second derivatives can then be represented as functions of *data value* as opposed to position (Figure 14). Kindlmann then restructures the graph in Figure 14 into a parametric plot showing the relationships between data value, first, and second derivative (Figure 15).

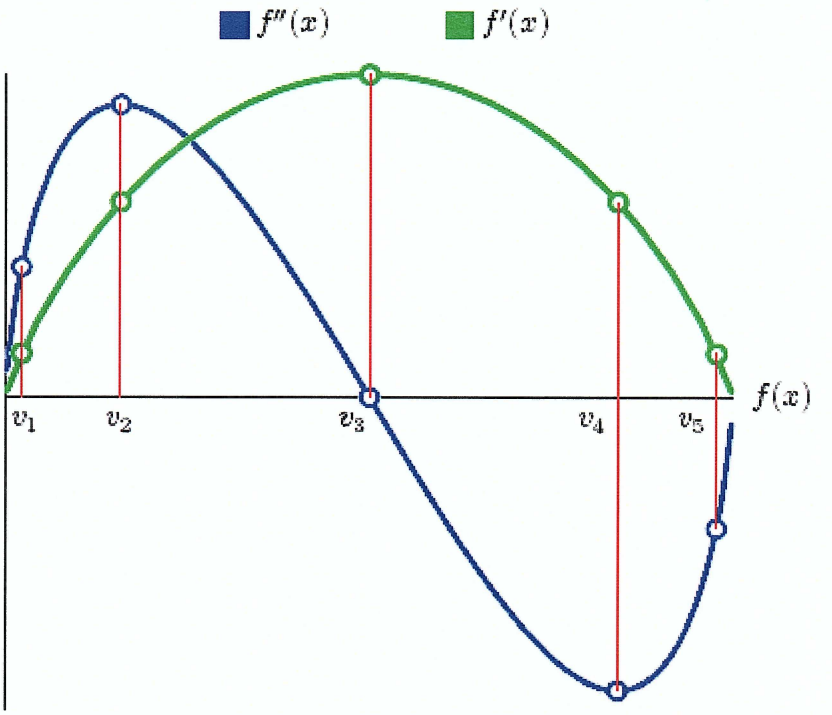


Figure 14: First and second derivative versus data value with derivative measures for data values v_1 through v_5 plotted (modified from [Kin99] Figure 3.5 (b)).

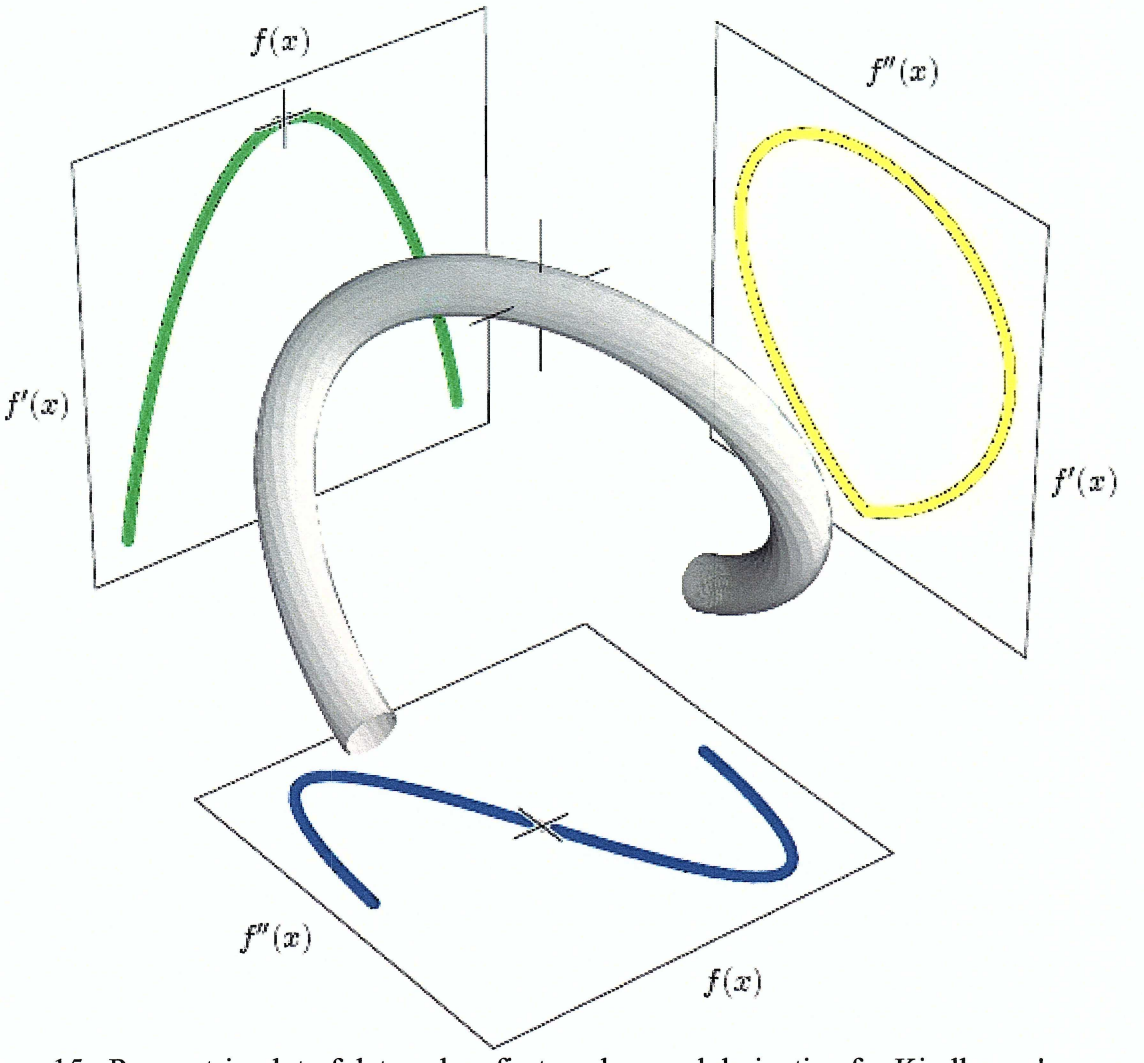


Figure 15: Parametric plot of data value, first, and second derivative for Kindlmann's boundary model (modified from [Kin99] Figure 3.8).

Recall from Figure 11 that the point in line Q directly in the center of the material boundary between materials *A* and *B* corresponds to the data value halfway between Val(*A*) and Val(*B*), the maximum first derivative, and the second derivative's zero value. Note that the cross-hair in Figure 15 represents this same point in parametric space. That is, since there exists a one-to-one relationship between data value and position *within a boundary*, any point on the parametric plot in Figure 15 can then be mapped to a position in that boundary. Further, if the boundary model holds for some volume data set, then the plot of data value, first, and second derivatives for each boundary voxel in the volume should produce a curve in parametric space similar to that shown in Figure 15. Once these curves are obtained, a

mapping can then be established from each data value in the volume to a position that corresponds to that value *in some material boundary*.

To produce this plot in parametric space Kindlmann uses a three-dimensional *histogram volume* to represent the frequency of value (f), first (f'), and second (f'') derivatives for all voxels in a volume. Specifically, let H be a histogram volume with dimensions (x, y, z) and let each unit of the histogram be termed a *bin*. Then each bin in the x dimension of the histogram represents a range of data values, each bin in the y dimension represents a range of first derivative values, and each bin in the z dimension represents a range of second derivative values. Each bin is assigned a value (initially set to zero) which represents the number of voxels in a volume that have f, f', and f'' within the ranges assigned that bin. Then for each voxel in the volume, Kindlmann determines f, f', and f'' of that voxel and records the maximum and minimum f'', and the maximum f' (assuming zero for the minimum first derivative) over all voxels. These extrema determine the range of f, f', and f'' represented by each bin of the histogram. A second pass of the volume is then made. Value, first, and second derivative are again measured for each voxel, a bin is determined that corresponds to that f, f', and f'' combination (based on the bins ranges), and the corresponding bin's value is incremented by one. The result then is an (x, y, z) grid of scalar values, where each bin's value represents how many voxels in the volume have a data value, first, and second derivative that fall within that bin's range. This histogram will then be used to determine a mapping from f to f' and f'', and thus a mapping from f to position (detailed in section 4.5).

For this thesis f' and f'' are obtained per voxel using central differencing and the Hessian method, respectively. Kindlmann shows that in a scalar field if $f(x) = \mathrm{Val}(A)$, then $f'(x)$ is the gradient magnitude at voxel A, and $f''(x)$ is the second directional derivative at voxel A. For the sake of notational convenience these terms are used interchangeably hereafter.

4.2 Histogram Volume

To visualize the histogram volume created using the method detailed above Kindlmann uses two-dimensional scatterplots such as the one shown in Figure 16.

Kindlmann assigns a gray level to each bin of the histogram based on the value of that bin (i.e. the number of voxels in the volume with f, f', and f'' corresponding to that bin's range) and shows two-dimensional projections of f and f' (as shown in Figure 16), or f and f'' (not shown). Note that four curves (such as the one graphed in Figure 15) are plainly visible in Figure 16. This indicates that Kindlmann's boundary model can accurately model the change in data value of voxels within material boundaries. In particular, each curve in Figure 16 represents a unique boundary between two materials, and the "peak" of each curve in Figure 16, that is, the data value where f' becomes a maximum, represents the zero position for that curve (the center of the material boundary).

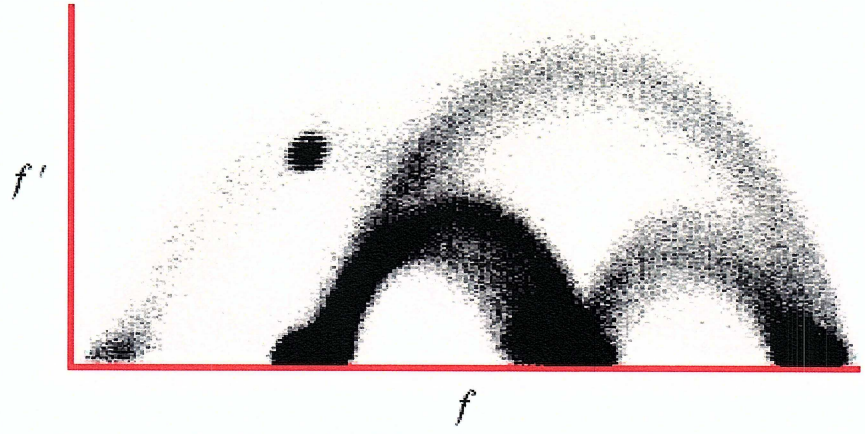


Figure 16: Two-dimensional scatterplot of the human tooth data set using Kindlmann's gray scale histogram visualization method.

As the histogram volume is later used in Kindlmann's method to determine a mapping from data value to position, the structure of the histogram can have a significant impact on the level of specificity of the mapping generated from it. Of particular significance is size of the histogram volume's dimensions, and the choice of f, f', and f'' ranges represented by each bin in the histogram. If the dimensions of the histogram volume are too small, each histogram bin will represent a relatively large range of f, f', and f'' values. This effectively reduces the resolution with which boundary curves can be specified, and thus diminishes the amount of information that the histogram volume can contain about each boundary curve. This, in turn, reduces the specificity of the mapping from value to position later derived from the histogram volume. In his thesis Kindlmann studies a variety of histogram volume

dimension sizes for different volume data sets, and notes subtle differences in results based on these dimension sizes. Along similar lines, the range of values in f, f', and f'' represented by each histogram bin also has a significant effect of the amount of information the histogram volume can contain about each boundary curve. The larger the range of values in f, f', and f'' represented by each bin, the less information the histogram volume can hold about the structure of the boundary curves. However, the smaller the range of values represented by each bin, the larger the histogram dimensions must be in order to contain every f, f', and f'' combination. Kindlmann's solution to this problem is to first determine the range of all possible first and second derivative values within the data set, and then select a subrange of these values by hand – a method which he declares in need of further research.

4.3 Linear Histogram Bin Scaling

In an attempt to automate the process of determining optimal f, f', and f'' value ranges per histogram bin, a new method for histogram volume creation is proposed. First, the histogram volume is assumed to have dimensions (256, 256, 256). These dimensions were found to work well with 8-bit data as each bin representing a data value range can be given a range of one data value (since 8-bit data ranges [0, 255]), and since f' and f'' are derived from data values with 8-bit precision it is natural to assume to storing these measurements with 8-bit precision results in nominal information loss. With histogram volume dimensions chosen, value ranges must now be selected for each bin. As mentioned previously, when using 8-bit data and 256 bins in the data value dimension, bins can be assigned an f range of one. Now assume (safely) that the minimum f' of all voxels in the data set is zero, and the maximum f', with value f'_{max} is determined by measuring the first derivative for each voxel in the data set. It is then possible to create *another* histogram of first derivatives alone. Let H' be a one-dimensional histogram of dimension n where each bin of H' represents a range of first derivative values. In particular, if bins are labeled b_1 through b_n then bin b_i represents the range $[(i-1)(f'_{max}/n),\ i \cdot (f'_{max}/n)]$. The algorithm then calculates f' for each voxel in the data set and increments the appropriate bin. The next step is to determine a subrange $[f'_{small}, f'_{large}]$ of $[0, f'_{max}]$ where $0 \le f'_{small} < f'_{large} \le f'_{max}$ such that at most p'_{cut} percent of all boundary voxels fall outside the range $[f'_{small}, f'_{large}]$.

That is, the algorithm must determine a range of f' values that is broad enough to capture *most* of the f' values found in boundary voxels, but narrow enough to allow each bin of the *histogram volume* to represent a small range of f' values (increasing the information content of the volume histogram). To determine this range, the algorithm estimates the number of boundary voxels *NumBV* in the volume by counting all voxels with non-zero gradient magnitude. It then determines how many boundary voxels can fall outside either end of the subrange by calculating $NumCutBV = 0.5 \cdot (100 \cdot p_{cut}) \cdot NumBV$. The algorithm then starts at bin b_1 and sums the value stored in each successive bin until the sum of bin values exceeds *NumCutBV*. At this point the bin's index bin_{small} is recorded and a similar process determines bin index bin_{large} by starting at bin b_n and working backwards. The minimum f' value of the range represented in bin bin_{small} is then recorded, as is the maximum f' value of the range represented in bin bin_{large}. Specifically, $f'_{SmallCut} = (bin_{small} - 1)(f'_{max} / n)$, and $f'_{LargeCut} = (bin_{large})(f'_{max} / n)$. The subrange $[f'_{SmallCut}, f'_{LargeCut}]$ of f' is then divided equally into the 256 f' ranges of the *histogram volume* such that voxels with f' values less than $f'_{SmallCut}$ or greater than $f'_{LargeCut}$ are clamped to f' bins 0 and 255 respectively in the histogram volume. A similar procedure is then used to determine $[f''_{SmallCut}, f''_{LargeCut}]$ which is divided amongst the 256 f'' ranges in a similar fashion.

As an example, consider an 8-bit volume data set with dimensions (64, 64, 64). The algorithm first determines the minimum and maximum f'' values for all voxels in the data set -2000 and 1500, respectively. Similarly, the minimum and maximum f' values are determined to be 0 and 84.5 respectively. (*sidenote – using 8-bit data and standard central differences method for gradient approximation yields a maximum measured gradient magnitude of root(3) * 127.5 = 167.799...). To determine the cutoff points for valuable f' values, the algorithm first creates a one-dimensional histogram of $n = 2048$ bins. Let $f'_{BinRange} = 84.5/2048 \approx 0.0413$. Then one-dimensional bins y_1 through y_{2047} are assigned f' ranges $\{ [0, f'_{BinRange}), [f'_{BinRange}, 2 \cdot f'_{BinRange}), \ldots, [2047 \cdot f'_{BinRange}, 84.5] \}$. For each voxel in the volume, f' is calculated and the counter in the appropriate bin is incremented. Once all voxels have incremented some one-dimensional bin, a user specified parameter p_{cut} is used to determine the maximum number of boundary voxels to be allowed outside the final f' represented by the histogram volume. If p'_{cut} is set to 0.001, then at most one in a thousand

of all boundary voxels will be allowed to fall outside this range (and thus must be clamped to the first or last bin). Assume the volume contains 8000 boundary voxels (voxels with non-zero first derivative). Then $NumCutBV = 4$, and the algorithm starts at one-dimensional bin y_1 and sums each successive bin's counter until the sum reaches or exceeds $NumCutBV$. This bin index is recorded as bin_{small}. Similarly, the algorithm starts at one-dimensional bin y_{2047} and sums each bin's counter going backwards until the sum reaches or exceeds $NumCutBV$. This bin index is recorded as bin_{large}. Assume $bin_{small} = 25$ and $bin_{large} = 1978$. Then the f' range represented by these bins and all bins between them is $[(bin_{small} - 1) \cdot f'_{BinRange}, bin_{large} \cdot f'_{BinRange}] \approx [(25\text{-}1) \cdot 0.0413, 1978 \cdot 0.0413] \approx [0.99, 81.6]$. Then if the *volume histogram* has dimensions (256, 256, 256), every volume histogram bin b_i with position (a, b, c) in the volume histogram contains the count of all voxels in the volume such that the data value of the voxel is a, the first derivative of the voxel is in the range $[(b - 1) \cdot ((81.6 - 0.99)/256), b \cdot ((81.6 - 0.99)/256)]$, and the second derivative is in a range $[(c - 1) \cdot f''_{BinInterval}, c \cdot f''_{BinInterval}]$ where $f''_{BinInterval}$ is determined in a method similar to that described above for f' (using a parameter p''_{cut} analogous to p'_{cut} with default value 0.001).

4.4 Histogram Volume Visualization

The benefit of using the method for histogram scaling is that the p_{cut} parameter allows the user to judiciously scale the histogram to optimize the "fit" of the volume data's value and derivative measures to the histogram without losing significant information to clamping effects. This is done visually with the aid of a histogram visualization tool developed specifically for this method. While Kindlmann represents volume histograms using two-dimensional gray scale images, the application developed in this research represents volume histogram structure by considering the histogram itself a volume data set and rendering a three-dimensional image. Additionally, rather than using a gray scale to denote bin frequency, a color scheme was developed (based on the HSV color space) that can be easily modified by the user to aid in visualization of boundary curves in the histogram volume.

The histogram visualization application begins by calculating f, f', and f'' for each voxel in the volume data set. The linear histogram bin scaling method described above is then run with a default p_{cut} value of 0.0001, essentially setting the volume histogram bins to

ranges that allow 99.99% of all boundary voxels to be represented in the histogram volume without clamping. After bin scaling constants have been determined, each voxel increments a bin in the histogram volume such that each histogram volume bin contains a count of the number of voxels in the f, f', and f'' range that bin represents. Before visualization, each of the histogram volume's bin's counts is scaled to [0, 1] based on two scaling parameters, S_{min} and S_{max} according to the following equation:

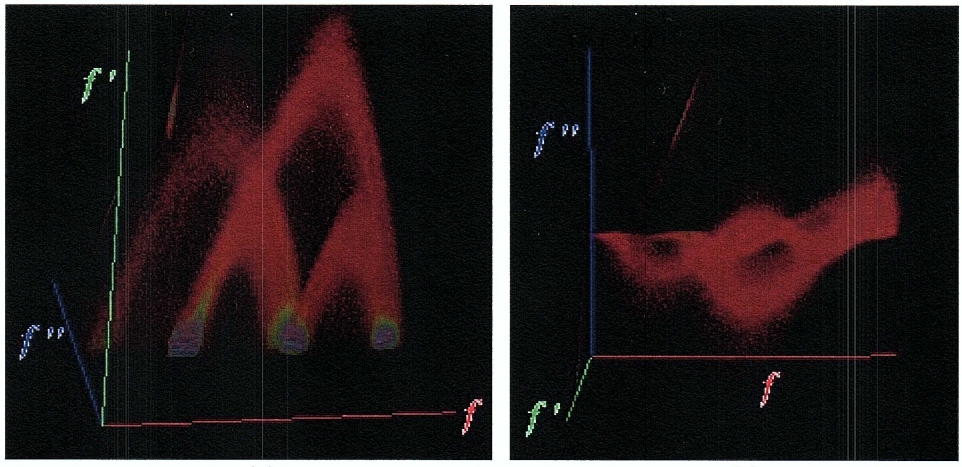
$$Hits_{scaled} = \begin{cases} 0 & \text{if } Hits_{unscaled} < S_{min} \\ 1 & \text{if } Hits_{unscaled} > S_{max} \\ \dfrac{H_{unscaled}}{S_{max} - S_{min}} & \text{otherwise} \end{cases} \quad \text{(Equation 5)}$$

where $Hits_{unscaled}$ is the voxel count for the given bin, and $S_{min} = 0$ and $S_{max} = 200$ by default. These scaled values, in addition to the minimum opacity parameter $Opac_{min}$ (with range [0, 1]), are then used to determine the color and opacity for each bin in the final rendering according to the formula:

$$Color = \text{HSV}(Hits_{scaled})$$

$$Opacity = \begin{cases} 0 & \text{if } Hits_{scaled} = 0 \\ Opac_{min} + (1 - Opac_{min}) * Hits_{scaled} & \text{otherwise} \end{cases} \quad \text{(Equation 6)}$$

where HSV maps [0, 1] to an appropriate hue (color) in the HSV color space, and saturation and value (brightness) are set to a constant maximum value. By Equation 6, each bin in the volume histogram is assigned a *Color* unique to the scaled voxel count it stores. Additionally, each bin is assigned an *Opacity* in the range [0, 1] (default 0.1) such that bins with larger scaled voxel counts are rendered more opaque than voxels with smaller scaled voxel counts, and bins with zero scaled voxel counts are rendered fully transparent (i.e. not rendered). Thus this coloring scheme assigns a unique color and opacity to each set of volume histogram bins with similar frequency in the volume data. Figure 17 shows the initial histogram volume visualization created using the default parameters on a data set of a human tooth.



(a) (b)

Figure 17: Histogram volume visualization of a human tooth using default parameters $Opac_{min} = 0.1$, $p_{cut} = 0.001$, $S_{min} = 0$, and $S_{max} = 50$.

Note in Figure 17 (a) that there are four distinct curves similar in structure to the curve graphed in parametric space in Figure 15. Each of these curves then represents a set of boundary voxels within the volume that form one distinct boundary between two different materials. Areas of similar color in Figure 17 represent volume histogram bins with similar frequency. The four "blobs" aligned with the f axis in Figure 17 (a) represent the four materials that compose the compose the boundaries in the volume data. That is, due to noise in the data set and variance of measured value within materials, areas of high frequency and low f' show up in the histogram for each material in the volume. Specifically, Figure 18 shows areas in the volume histogram visualization representing each of the four materials and the curves between these materials.

While the default parameters for volume histogram visualization have proven sufficient for visually distinguishing the curves in all volume data sets used in this research, visualization can be enhanced by manually tweaking the parameters S_{min}, S_{max}, and $Opac_{min}$. Note that some boundaries in the Figure 18 are more easily distinguished than others. In particular, the curve c(1,3) (the curve from material 1 to material 3) is less distinguishable than curve c(2, 3). Curve c(1,3) is more sparse than c(2, 3) because the boundary between material 2 and material 3 contains more voxels than the boundary between materials 1 and 2. The clamping effect of parameter S_{max} then allows the user to view these sparse curves in conjunction with denser curves. Figure 19 illustrates the results of

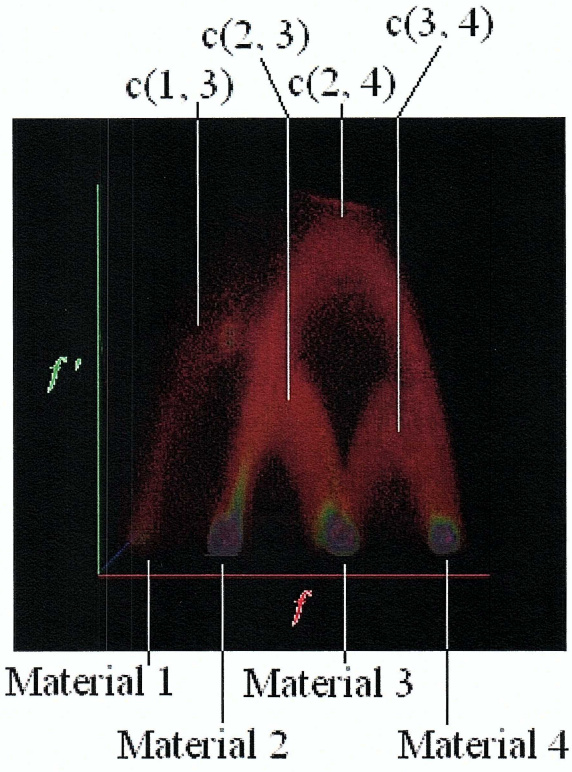
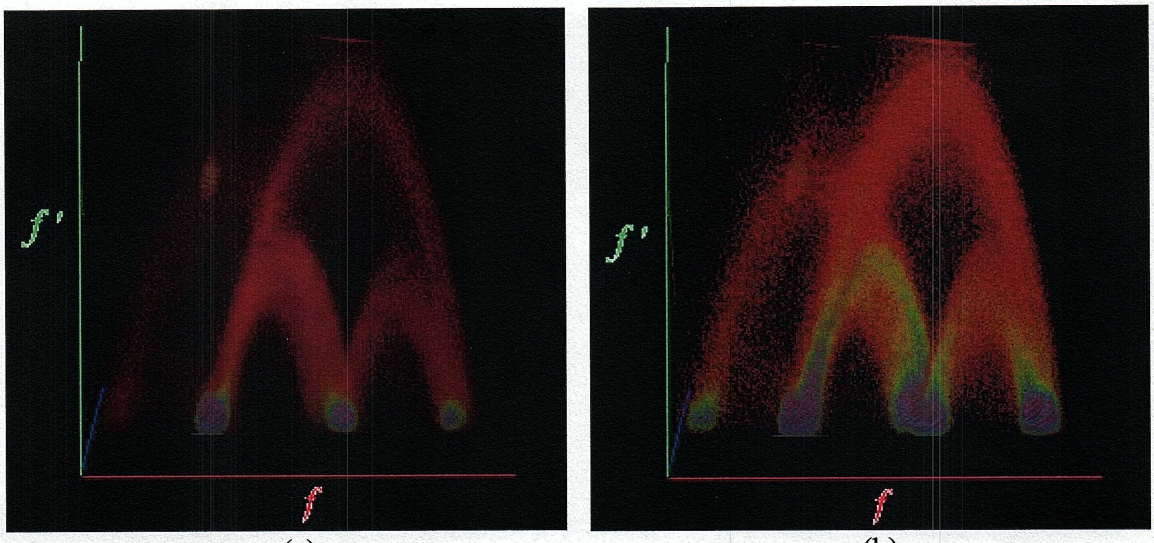


Figure 18: Histogram volume of the human tooth data set using default parameters.

modifying S_{max} from a large value to a smaller value; note the improved visibility of curve c(1,3). The S_{min} parameter then allows the user to exclude lower frequency bins from the visualization, effectively reducing noise in the histogram volume and making curve structure more evident, while the $Opac_{min}$ parameter can be increased to make sparse curves more visible, or decreased to make the internal structure of high frequency curves more visible behind low frequency noise.

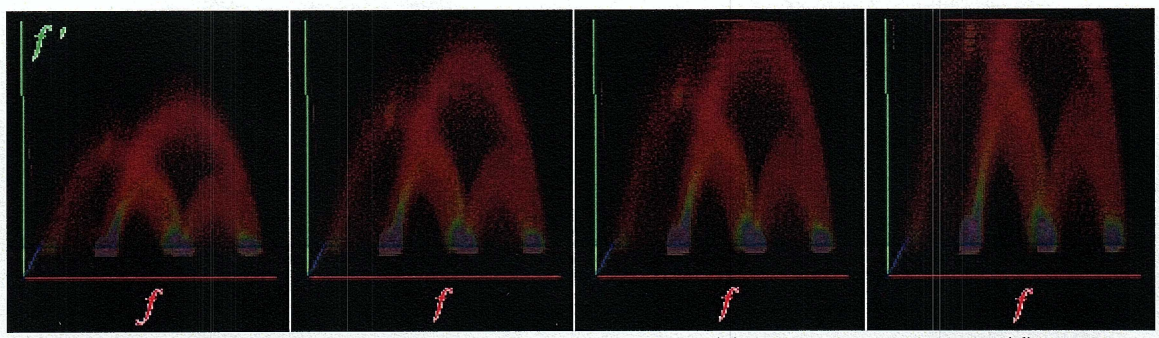
The goal of the histogram volume visualization tool is two-fold. First, it gives the user an intuitive understanding of the structure of the boundary curves within parametric space. More importantly, it serves as a visual aid in determining appropriate histogram bin scaling parameters p'_{cut} and p''_{cut}. That is, it allows the user to visually inspect the success of the linear histogram bin scaling described above in "fitting" the curves to the histogram such that the amount of information the histogram volume contains about the boundary curves is maximized. Figure 20 shows the results of linear histogram bin scaling with various p'_{cut} settings. Notice in Figure 20 that increasing p'_{cut} has the effect of "stretching" the curves in



(a) (b)

Figure 19: Histogram volume of the human tooth data set using varying parameters. Image (a) was produced using $S_{min} = 0$, $S_{max} = 200$, and $Opac_{min} = 10$. Image (b) highlights the internal structure of the boundary curves by modifying the S_{max} parameter to a value of 10.

the f' direction. Using a p'_{cut} of zero (Figure 20 (a)) shows the result of scaling f' bin ranges based solely on the maximum f' in the data set. Figure 20 (d) shows the result of linear bin scaling such that 0.01% of f' values get clamped to the largest f' bins. Notice, however, that this relatively high setting does not provide a good fit for the tooth data set as a large number of the f' values in the c(2, 4) curve get clamped to the largest f' bins. The visualization tool allows the user to see this and adjust the parameters accordingly. A similar process allows the user to adjust the p''_{cut} parameter to produce a good fit for the histogram volume on the f'' axis.



(a) (b) (c) (d)

Figure 20: Histogram volume visualization with p'_{cut} settings of 0, 0.0001, 0.001, 0.01, respectively.

4.5 Mapping Data Value to Position

Recall that Kindlmann's goal is to determine a mapping from data value to position in order to create a transfer function for the visualization of boundaries within a volume data set. The histogram volume in conjunction with Kindlmann's boundary model serve as the key to developing this mapping. The boundary model uses the equation

$$v = f(x) = v_{min} + (v_{max} - v_{min}) \frac{1 + \mathrm{erf}(\frac{x}{\sigma\sqrt{2}})}{2} \quad \text{(Equation 1)}$$

to map the position of a voxel within a material boundary to a value ranging from v_{min} to v_{max}, the values of the material which compose the boundary. As mentioned previously, σ in Equation 1 represents the voxel width of the material boundary, which is assumed to be constant throughout the boundary. Note, then that sigma determines the overall shape of the boundary's curve in the histogram volume. A thick boundary (with large σ) will have a relatively slow increase in data value as voxels are traversed through the boundary. This results in a relatively small maximum f', and thus minimum and maximum f'' values which are relatively close to zero.

Also note that Equation 1 makes use of the error function erf, defined as

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\Pi}} \int_0^x e^{-t^2}\, dt \quad \text{(Equation 2)}$$

As a result of the special derivative properties of e^x, Kindlmann is able to determine a simple formula (Equation 7) that relates f', f'', position x, and boundary width σ.

$$\frac{f''(x)}{f'(x)} = \frac{-x}{\sigma^2} \quad \text{(Equation 7)}$$

Equation 7 implies that the position of a voxel within a boundary can be determined from the first and second derivatives of data value of the voxel, and the width σ of the boundary the voxel is within. Again as a result of the derivative properties of e^x, Kindlmann is able to derive σ in terms of $f'(0)$ and $f''(-\sigma)$ or $f''(\sigma)$ in Equation 8. Recall from Figure 12 that in

any material boundary modeled by Kindlmann's boundary model, f' reaches a maximum at position zero, and f'' reaches a maximum and minimum at position -σ and σ , respectively.

$$\sigma = \frac{f'(0)}{\sqrt{e}\cdot f''(-\sigma)} = -\frac{f'(0)}{\sqrt{e}\cdot f''(\sigma)} \quad \text{(Equation 8)}$$

Equation 8 can then be interpreted as

$$\sigma = \frac{f'_{max}}{\sqrt{e}\cdot f''_{max}} = -\frac{f'_{max}}{\sqrt{e}\cdot f''_{min}} = \frac{2\sqrt{e}\cdot f'_{max}}{f''_{max} - f''_{min}} \quad \text{(Equation 9)}$$

where the last term in Equation 9 expresses σ in terms of the maximum first derivative and the minimum and maximum second derivative. To determine σ Kindlmann's method simply finds the maximum f', and the minimum and maximum f'' in the data set, and then inserts these values into the last term of Equation 9.

Using Equation 7 and Equation 9 it is then possible to calculate the position of a voxel within a boundary knowing only the first and second derivatives of that voxel. This provides a mapping from f' and f'' to position, thus all that is required to complete a mapping of data value to position is a mapping from data value to f' and f''. This final mapping is produced by observing f' and f'' at each data value in the histogram volume. Kindlmann defines $g(v)$ as the average f' over all positions x at which $f(x) = v$, and similarly defines $h(v)$ as the average f'' at value v. To obtain $g(v)$ a slice of the histogram volume for some value v is analyzed, and the centroid of the histogram bins in that slice is determined. The position of the centroid along the f' axis is recorded as $g(v)$, and the position of the centroid along the f'' axis is recorded as $h(v)$ (Figure 21).

Thus for each *data value* v in the histogram, $g(v)$ and $h(v)$ represent the *average* first and second derivative measures of all voxels with data value v. This then provides the desired mapping from data value to f' and f'', which can in turn be mapped to position using Equation 7 and Equation 9. Kindlmann defines the position function $p(v)$ that performs this mapping as

$$p(v) = \frac{-\sigma^2 h(v)}{g(v)} \quad \text{(Equation 10)}$$

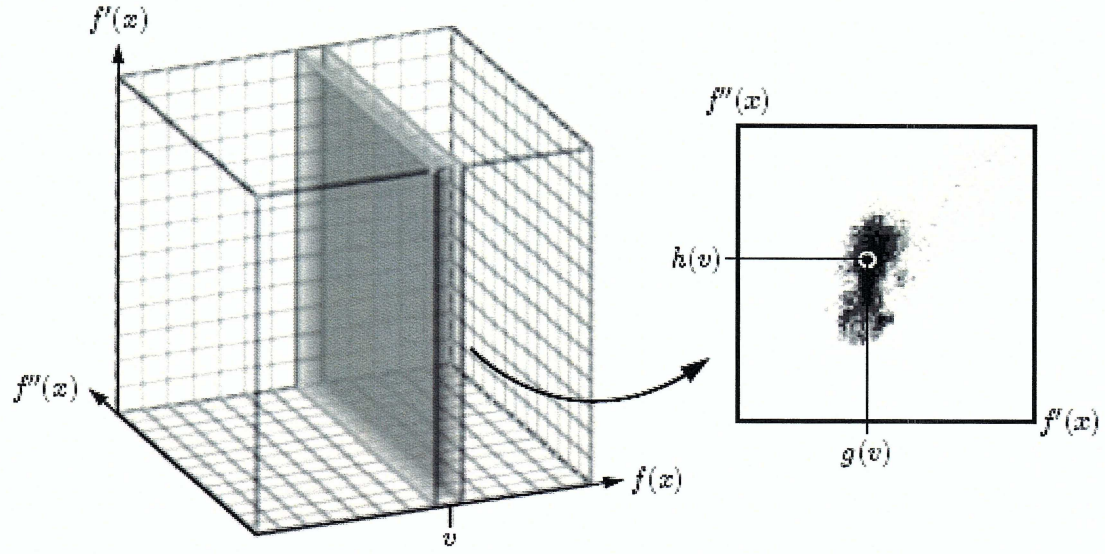


Figure 21: Value aligned slice of the histogram volume used to approximate derivative measures at that value (from [Kin99] Figure 5.3).

Then with $h(v)$ and $g(v)$ determined via the histogram volume, Equation 10 can be used to map each voxel to a position in a material boundary based solely on the voxel's value.

Note that $h(v)$ and $g(v)$ are essentially approximations of the f' and f'' measurements of every voxel at a position in an ideal boundary that corresponds to value v. As a means to further increase the accuracy of this approximation, Kindlmann extends Equation 8 to calculate the average f' and f'' measurements of all voxels at a particular data value *and* first derivative using Equation 11

$$p(v,g) = \frac{-\sigma^2 h(v,g)}{g} \quad \text{(Equation 11)}$$

where $p(v, g)$ is a position function mapping value v and first derivative measure g to a position in a material boundary, and $h(v, g)$ is determined by find the the average f' for all voxels with data value v and first derivative g. That is, rather than finding the centroid of a two-dimensional slice of the histogram volume, $h(v, g)$ is determined by looking at a "sliver" of the histogram volume with value v and first derivative g (Figure 22) and determining the average f'' value.

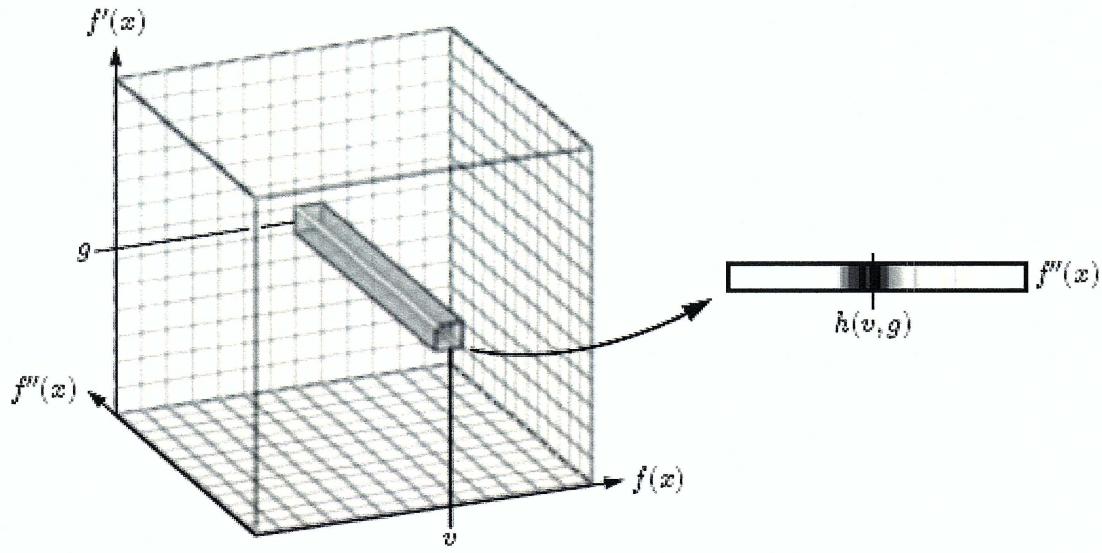


Figure 22: Slice of the histogram volume at a given data value and first derivative pair.

4.6 Boundary Specific Histogram Volumes

While Kindlmann's position functions $p(v)$ and $p(v, g)$ work well in determining position in some data sets, both functions are based on a critical assumption that does not hold true in all cases. Using $g(v)$ and $h(v)$ to approximate f' and f'' makes the implicit assumption that there exists a one-to-one mapping from data value to first and second derivative measures, and thus a one-to-one mapping from data value to position. While it is true that there exists a one-to-one mapping from data value to position *within a single boundary*, each boundary will have a *different* data value at which the position in that boundary is zero. Moreover, if one differentiates boundaries by the materials that compose each boundary, then if two boundaries have a position of zero that maps to the same data value these two boundaries are between the same materials and thus form the same boundary. Figure 22 illustrates this concept. First consider the volume shown in the upper section of Figure 22 (a) which consists of three distinct materials A, B, and C. The graph in the lower section of the same figure shows the change in data value along line segment Q. As seen previously in Figure 10 (b), data values change gradually over in the boundary regions (indicated in green) and remain fairly constant within each material. Note from the graph in Figure 22 (a) that materials A, B, and C have successively large data value. The data

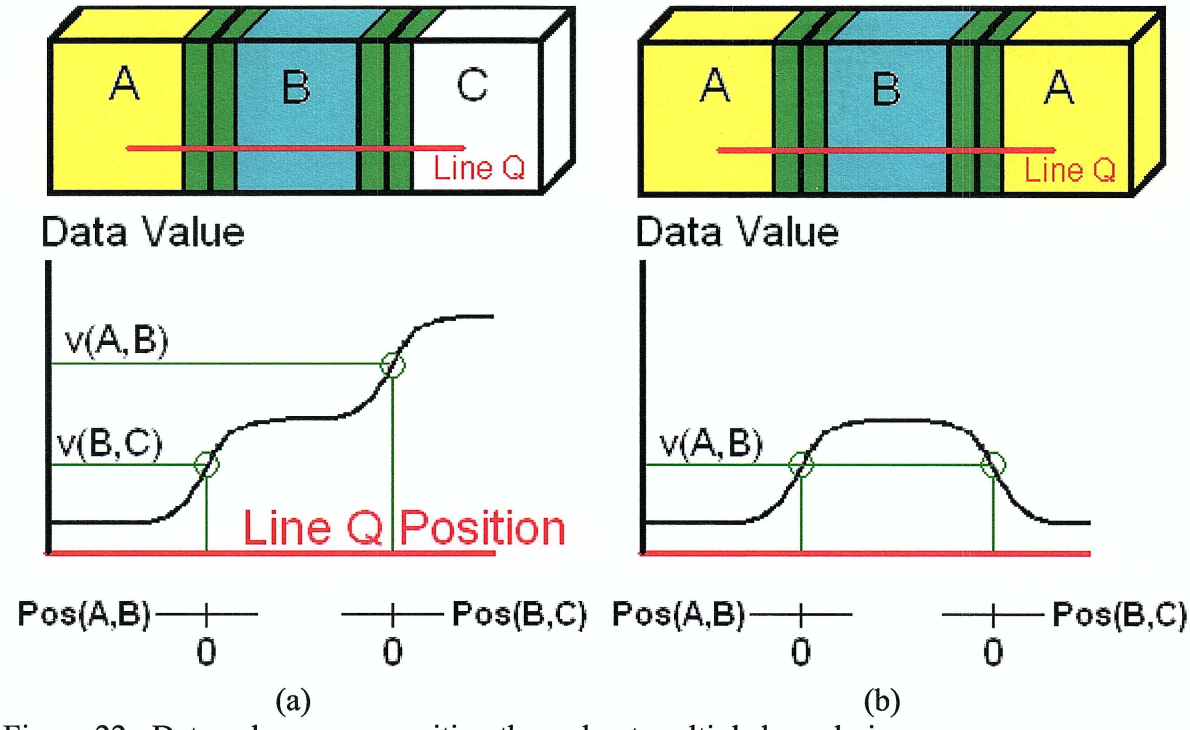
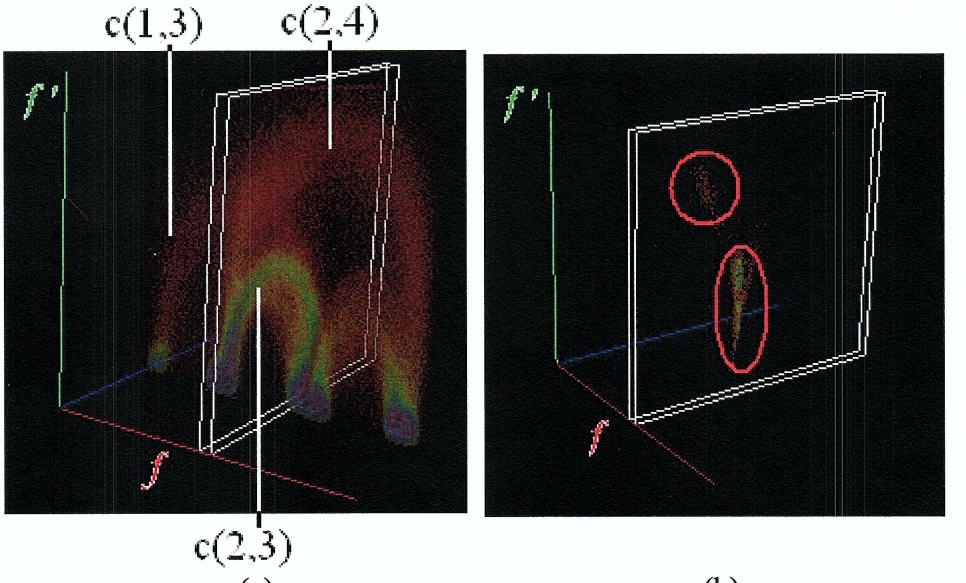


Figure 22: Data value versus position throughout multiple boundaries.

values corresponding to the *center* of the material boundaries between each of the materials is indicated on the Data Value axis by v(A,B) and v(B,C) for the boundaries between materials A and B, and B and C, respectively. These data values are different for each of the two material boundaries, and thus the position functions for each of these boundaries (represented as Pos(A,B) and Pos(B,C)) each have a zero position that maps to a different data value. Figure 22 (b) shows that two spatially distinct boundaries between the *same* two materials will have position functions which map position zero to the same data value. However, in both circumstances illustrated in Figure 22 (a) and (b), neither boundary will map any non-zero position to the same data value. For example, in the volume of Figure 22 (a), a data value between v(A,B) and v(A,C) will be mapped to a positive position in the position function Pos(A,B), but will be mapped to a negative position in Pos(B,C).

As a consequence of the assumption that there exists a one-to-one mapping from data value to position, the method by which Kindlmann's method calculates $g(v)$ and $h(v)$ is prone to making poor f' and f'' approximations in specific cases. Consider the histogram volume of the human tooth shown in Figure 17. Any data value slice of the histogram volume has

high probability of intersecting not one, but multiple boundary curves. Finding the centroid of all bins in this slice to approximate f' and f'' at this data value then "pulls" the approximation towards the densest boundary curve at that value.



(a) (b)

Figure 23: Data value aligned slice of the histogram volume in the human tooth data set.

An example of a data set in which $g(v)$ and $h(v)$ yield inaccurate approximations can be seen in Figure 23 above. The white borders indicate a slice of the histogram volume at a data value v. Notice in Figure 23 (a) that the slice intersects three curves: c(1,3), c(2,3), and c(2,4). Figure 23 (b) shows only those bins within the given slice. The centroid of the bins within this slice will then be "pulled" in the direction of the bins within curve c(2,3) as it represents the densest region (bins with highest voxel count) in the slice. That is, $g(v)$ and $h(v)$ will more accurately represent the f' and f'' values of the boundary between materials 2 and 3 at the given data value than it will those of the boundaries between material 1 and 3 or 2 and 4. More importantly, the position function $p(v)$, which uses $g(v)$ and $h(v)$ in its calculation, will map data value v to a position in terms of the boundary represented by curve c(2,3). Then any voxel with data value v that is *not* in the the boundary between materials 2 and 3 will be assigned an erroneous position. Specifically, $p(v)$ will map *every* voxel with data value v to a position in terms of the boundary represented by curve c(2,3), regardless of the boundary to which the voxel belongs.

In general, $g(v)$ and $h(v)$ will result in poor f' and f'' approximations in any data set where boundary curves in the histogram volume "overlap" at some data value v. That is, if a data value slice at v intersects multiple curves, the centroid of that slice will be closer to position where the densest curve intersects the slice, and thus the approximations of f' and f'' will more closely represent the derivative measures of the densest curve (leading to poor derivative approximations at value v for voxels belonging to every other intersecting boundary curve). The probability of overlap can be reduced somewhat using the position function $p(v, g)$ in Equation 11, as this position function looks at slivers of the histogram volume, rather than slices. These slivers of data value and first derivative, however, still frequently intersect multiple curves, and thus $p(v, g)$ is also subject to producing poor derivative approximations.

To avoid these approximation errors, the method presented in this thesis attempts to calculate $g(v)$ and $h(v)$ for *each material boundary*. This requires a separate histogram volume for *each* material boundary, and thus requires a method by which voxels can be assigned to a particular boundary. This thesis considers the distinguishing feature of individual material boundaries to be the materials which compose them. Referring back to Figure 22 (a), by this consideration the two regions of the volume shown in green represent two distinct material boundaries as each is between two different materials (A and B versus B and C). Whereas the material boundaries shown in Figure 22 (b) represent two instances of the same boundary, as each boundary is between the same two materials (A and B). Classifying which boundary a voxel belongs to is then a matter of determining which materials the voxel is *spatially between* in the volume data set. The method proposed accomplishes this classification by the use of the gradient vector for each voxel.

The first step of the algorithm is to determine which voxels are to be considered boundary voxels. Recall from Chapter 3 that classification of boundary voxels in accomplished during segmentation by the use of the gradient magnitude (first derivative) threshold parameter GM_{thresh}, such that all voxels with gradient magnitude greater than GM_{thresh} are considered boundary voxels (voxels that correspond to multiple materials in the real-world object). Then for each boundary voxel a *gradient vector of data value*, vec_G, is determined such that vec_G indicates the spatial direction within the volume of greatest data

value change. As Kindlmann's boundary model assumes that data values change monotonically within boundary regions, the gradient vector of a boundary voxel naturally "points" in the direction of a material that composes the boundary. Additionally, the opposite direction of the gradient vector, $-vec_G$, indicates the spatial direction of the other material which composes the boundary. More intuitively, if one considers the boundary region between two materials in a volume to be a surface, then the boundary model predicts that the gradient vector (and negative gradient vector) at any point of the surface will be orthogonal to the surface, and will thus "point" in the direction of one of the materials which the boundary surface is between. The algorithm then follows the gradient vector for each boundary voxel until a material voxel (classified during region growing) is reached. The material to which this material voxel is assigned (during segmentation) is recorded, and an analogous action is taken in the negative gradient vector direction. On completion of this step each boundary voxel has been assigned two distinct material regions (one each in the vec_G, and $-vec_G$ direction) that represent the two materials which compose the boundary to which the voxel belongs. For each combination of two material regions a *boundary index* is assigned which uniquely identifies that boundary, and all boundary voxels found to be between two materials are assigned an appropriate boundary index.

Figure 24 below illustrates this process. Returning to the cross section of the human tooth data set (Figure 7), Figure 24 shows boundary regions in white, and three different materials in red, yellow, and blue. Consider voxel A on the boundary between the red and yellow materials. The gradient vector of voxel A points in the direction of greatest data value change (shown as a black line from the voxel), which is a vector orthogonal to the boundary between red and yellow. Following in the gradient (and negative gradient) vector's direction then leads to a material. The algorithm detects that voxel A is between red and yellow, and that voxel B is between yellow and blue. As these two combinations contain different materials, two unique boundary indices, $Bound_A$ and $Bound_B$ are created, such that every boundary voxel between the red and yellow material is assigned to boundary $Bound_A$, and every boundary voxel found to be between the yellow and blue materials is assigned to boundary $Bound_B$.

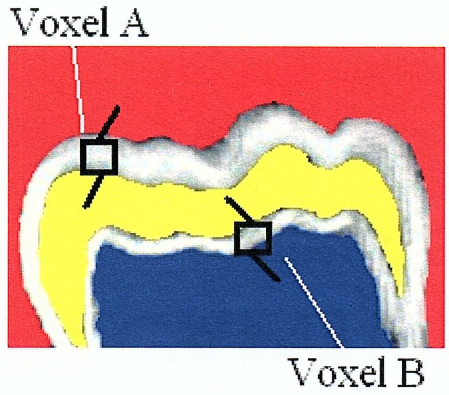


Figure 24: Determining material boundaries through gradient vectors.

Once every voxel has been assigned a boundary index it is possible to construct a histogram volume for *each* boundary in the volume by considering only voxels within a single boundary index (and thus a single boundary). By making derivative approximations along slices of data value in the individual histogram volumes the problem of overlap (multiple boundaries intersecting a single slice) is minimized. This allows for more accurate approximations of f' and f'' per boundary, and thus produces more accurate value to position mappings. These mappings are produced for each boundary using an extended version of Kindlmann's position formula:

$$p(v,b) = \frac{-\sigma^2 \, h(v,b)}{g(v,b)} \quad \text{(Equation 12)}$$

where $p(v, b)$ is the position function with respect to data value v and boundary index b. The terms $g(v, b)$ and $h(v, b)$ are determined by looking at a data value slice of the histogram volume constructed only from boundary voxels with boundary index b, and determining the f' and f'' positions of the centroid of bins in that slice. This method of determining position is identical to Kindlmann's position formula in Equation 12, with the exception that derivatives are approximated using a histogram volume constructed *only* from boundary voxels assigned a given boundary index (and thus all belong to the same material boundary). Assume there are B boundary indices (and thus B material boundaries) determined for a particular data set. The algorithm then generates B histogram volumes (one for each boundary index) and creates a position function for each boundary index which maps value to position within that boundary.

4.7 Mapping Position and Boundary Index to Optical Properties

Once a position function has been determined, the final stage in Kindlmann's method is to create a mapping from position to opacity. The goal of this mapping is to map voxels near the center of the a material boundary (those near position zero) to higher opacity in the hopes of making material boundaries in the final rendering more visible. Kindlmann produces this mapping using what he terms a *boundary emphasis function.* The boundary emphasis function $b(x)$ maps position x to an opacity in the range [0, 1]. Kindlmann's opacity function $\alpha(v) = b(p(v))$ then maps each data value to an opacity. A common structure for boundary emphasis functions is a linear ramp which assigns highest opacity to boundary voxels with position zero, and assigns opacity to non-zero positions in a symmetric and linearly decreasing fashion as shown in Figure 25.

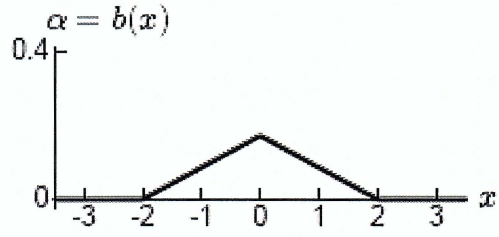


Figure 25: Kindlmann's opacity function in terms of position (from [Kin99] Figure 5.5).

Kindlmann allows the user to modify the boundary emphasis function manually to produce a variety of effects on the final rendered image. For example, a wider linear ramp (one that assigns non-zero opacity to more position measures) will produce thicker boundaries in the final rendered image, whereas a thinner ramp will assign higher opacity only to those voxels very near the material boundary center, and thus produce more accurate but thinner boundaries in the rendered image. The benefit of using boundary emphasis functions in conjunction with a position function for opacity assignment is that the user needn't understand the correlation between data value and opacity to specify a useful opacity function. Modification of the opacity function in terms of *position* is a much more intuitive means of modifying the final rendered image.

As one of the goals of this thesis is to reduce the amount of user input required in producing an informative transfer function, a simple method for the creation of initial boundary emphasis functions was developed based on a single parameter $Opac_{max}$. For each boundary index b the algorithm determines the minimum and maximum position attained by voxels assigned that boundary index, Pos^{b}_{Min} and Pos^{b}_{Max}, respectively. The "starting" position of the ramp Pos^{b}_{Start} is then determined as

$$Pos^{b}_{Start} = \frac{\min(Pos^{b}_{Max}, |Pos^{b}_{Min}|)}{2} \quad \text{(Equation 13)}$$

such that the range [- Pos^{b}_{Start} , Pos^{b}_{Start}] is centered at position zero and starts and ends the ramp at positions with a distance from zero that is one-half the minimum of the distance from zero to the maximum or the minimum position attained by a voxel in the given boundary. The peak of the ramp is at position zero with an assigned opacity of $Opac_{max}$ such that the entire opacity function for boundary index b, termed $Opacity^{b}(x)$, at position x is defined as

$$Opacity^{b}(x) = \begin{cases} 0 & \text{if } x < -Pos^{b}_{Start} \text{ or } x > Pos^{b}_{Start} \\ \dfrac{Opac_{Max}}{Pos^{b}_{Start}} \cdot x + Opac_{Max} & \text{if } -Pos^{b}_{Start} \leqslant x \leqslant 0 \\ \dfrac{Opac_{Max}}{-Pos^{b}_{Start}} \cdot x + Opac_{Max} & \text{if } 0 < x \leqslant Pos^{b}_{Start} \end{cases} \quad \text{(Equation 14)}$$

The opacity ramp defined by $Opacity^{b}(x)$ for each boundary index b can then assigns higher opacity to boundary voxels within the given boundary which are near the center of the material boundary.

A limitation of Kindlmann's opacity function is that it, by definition, only assigns opacity to each voxel, and thus all voxels are rendered the same color. As Kindlmann's method is unable to differentiate individual boundaries it is unable to color voxels belonging to each boundary differently. The method proposed in this thesis solves this problem through the assignment of a boundary index to each boundary voxel which identifies the boundary the voxel belongs to. With this ability to differentiate which boundary a voxel belongs to, it becomes possible to assign each voxel a color based on its boundary index. Coupled with the $Opacity^{b}(x)$ function defined for each boundary, this allows for creation of a mapping

Optical(*v, b*) from data value *v* and boundary index *b* to color and opacity, as shown in Equation 15.

$$Optical(v,b) = (HSV(b/B),\ Opacity^{b}(p(v,b))) \quad \text{(Equation 15)}$$

The function *HSV*(*b*/*B*) in Equation 15 maps [0,1] to an appropriate hue (color) in HSV color space (with maximum saturation and brightness), where *B* is the total number of boundary indices found in the data set. This allows the initial assignment of a unique color to each boundary in the volume, while the $Opacity^{b}(p(v,b))$ term assigns an appropriate opacity to each voxel with value *v* belonging to boundary *b* such that voxels near the center of the material boundary *b* are assigned higher opacity.

Once the *Optical*(*v, b*) function has been determined, it is then possible to assign a color and opacity to each voxel in the volume data set such that all voxels within the same boundary are rendered the same color, and voxels nearest the center of each material boundary are rendered more opaque. An additional advantage to specifying optical properties in this manner is that each boundary's optical properties can be easily changed by the user. Using Kindlmann's method any user-made modification to optical properties effects *all* voxels in the rendering, whereas the method presented in this thesis allows an individual boundary's color and opacity to be modified separately.

4.8 Algorithm Summary

The direct volume rendering method proposed in this thesis can be summarized as follows:

Global Histogram Volume

- Calculate f' and f'' for each voxel in the volume data set.
- Use linear histogram bin scaling along with parameters p'_{cut} and p''_{cut} to optimize the fit of the global histogram volume to the parametric plot of data value, f', and f'' combinations in the volume.
- (Optional) Allow the user to view a three-dimensional visualization of the histogram volume and manually change parameters p'_{cut} and p''_{cut} to obtain a histogram volume that optimizes information content.

Segmentation
- Determine boundary voxels based on each voxel's gradient magnitude.
- Determine all voxels which are in the interior of a material region.
- Merge all adjacent interior voxels into material regions
- Grow each region by including all non-boundary voxels with data values in a small range of the region's mean data value until no more voxels can be confidently included.
- Merge all adjacent regions with similar mean data values (based on parameter J_A).
- Merge all material regions with *very* similar mean data values, regardless of adjacency (based on parameter J_G).

Boundary Classification
- For each boundary voxel
  - Determine which two material regions the voxel is between by following the gradient and negative gradient vector.
  - Assign a boundary index based on the material regions that compose the boundary the voxel is within.

Transfer Function Generation
- For each boundary index b
  - Construct a histogram volume by analyzing only those voxels assigned to boundary index b.
  - Approximate f' and f'' at each data value by finding the position of the centroid in a slice of the histogram volume for b.
  - Derive from the f' and f'' approximations a position function $p(v, b)$ that maps each data value to a position within boundary b.
  - Using parameter $Opac_{max}$ determine an opacity function $Opacity^b(x)$ that maps position in boundary b to opacity in the final rendered image.
  - Determine a unique color for all voxels in boundary b based on the HSV color space.

Rendering
- For each voxel in the volume data set
  - If the voxel is a boundary voxel
    - Determine the data value v and the boundary index b of this voxel.
    - Assign optical properties $Optical(v, b)$ to the voxel.
  - Else
    - Assign completely transparent optical properties to the voxel
- Render each voxel according to its optical properties and apply the Phong shading model.

Chapter 5. Results

The results section below demonstrates the results obtained from three data sets commonly used in demonstrating DVR techniques: the human tooth CT, the engine block CT, and a portion of the Chapel Hill CT data set corresponding to a human head.

5.1 Human Tooth

One data set commonly used in demonstrating DVR methods is the human tooth CT scan. The tooth data set used in this research is an 8-bit volume with voxel dimensions (128, 128, 256). This data set is ideal for demonstrating the effectiveness of the proposed volume rendering method as it contains relatively little noise and has four distinct materials that compose four relatively uncomplicated boundaries. In addition, these boundaries frequently overlap in both data value and f' measures within the global histogram volume, allowing the expressiveness of using multiple histogram volumes to be seen. The section that follows not only shows the final rendered image, but illustrates each step of the proposed method as the algorithm is executed.

After calculation of f' and f'' for each voxel, the default values for parameters p'_{cut} and p''_{cut} (0.001 and 0.0003) are used in linear bin scaling to produce the global histogram volume shown in Figure 27. As with this data set, it has been found that the parametric plots of data value, f', and f'' for most volume data sets can be accurately fit within the global histogram volume using the default values for parameters p'_{cut} and p''_{cut}. Selection of the gradient magnitude thresholds GM_{mat} and GM_{bound} that determine which voxels are classified as interior material voxels and boundary voxels, respectively, is performed via an interactive method which shows the user the location of the two thresholds with respect to the global histogram volume. Figure 26 indicates the selected GM_{mat} and GM_{bound} thresholds of 10 and 50, respectively, as white planes in the histogram volume. Careful selection of these two thresholds is crucial to the success of later stages of the algorithm. If the GM_{bound} parameter is chosen too low such that material voxels are misclassified as boundary voxels, the segmentation section of the algorithm will not grow each region sufficiently, which ultimately leads to difficulty in determining which materials a boundary voxels is between.

On the other hand, setting GM_{bound} too high may result in classifying too few voxels as boundary voxels, and the boundaries rendered in the final stage of the algorithm can become sparse or "spotty". Similarly, as GM_{mat} is used to determine which voxels are within the interior of a material region, setting GM_{mat} too high risks classifying boundary voxels as interior voxels (which leads to improper merging of material regions during region growing), while setting GM_{mat} too low may yield too few interior voxels for the segmentation section of the algorithm to fully grow a region. Recall that the "blobs" of low f' in the histogram volume are the result of slight data value variance within a material, and each blob can be considered to correspond to material voxels in the volume. It has been found that using the histogram volume visualization to visually set the GM_{mat} threshold somewhere within the center of these blobs and the GM_{bound} parameter well above these blobs (but not so high that some boundary curve is fully below GM_{bound}) results in acceptably accurate classification of interior and boundary voxels.

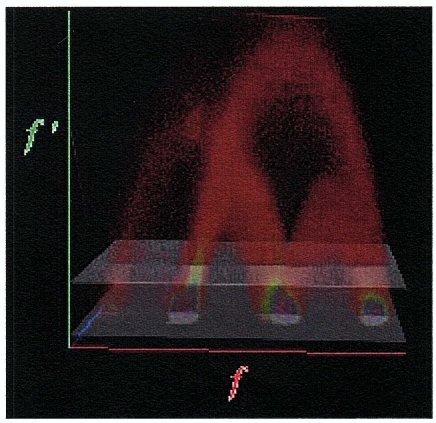
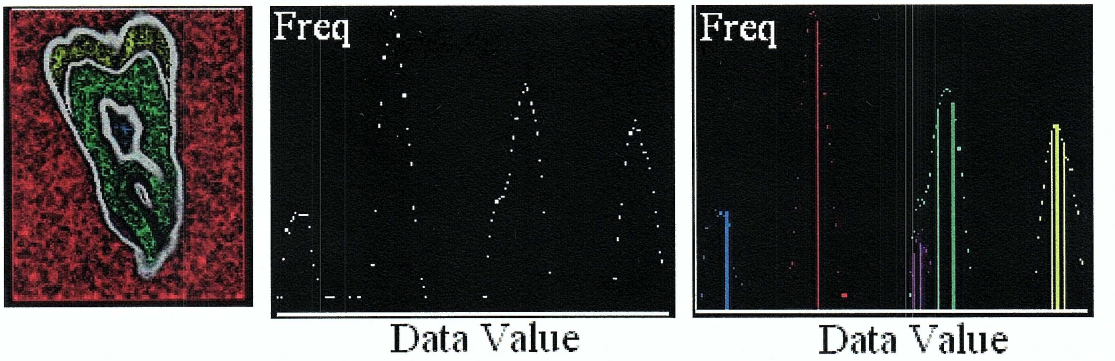


Figure 26: Histogram volume of the human tooth data set with parameters GM_{mat} and GM_{bound} represented by white planes.

The next stage of the algorithm is segmentation in order to identify material regions in the volume. All adjacent voxels with gradient magnitude (f') value less than GM_{mat} are grouped into material regions. Figure 27 (a) shows each initial material region in a cross section of the tooth as a separate color such that voxels in the interior of the air material region are colored red, enamel is yellow, dentin is green, and the root of the tooth is blue. Note that for each low f' material blob in the global histogram volume the algorithm has

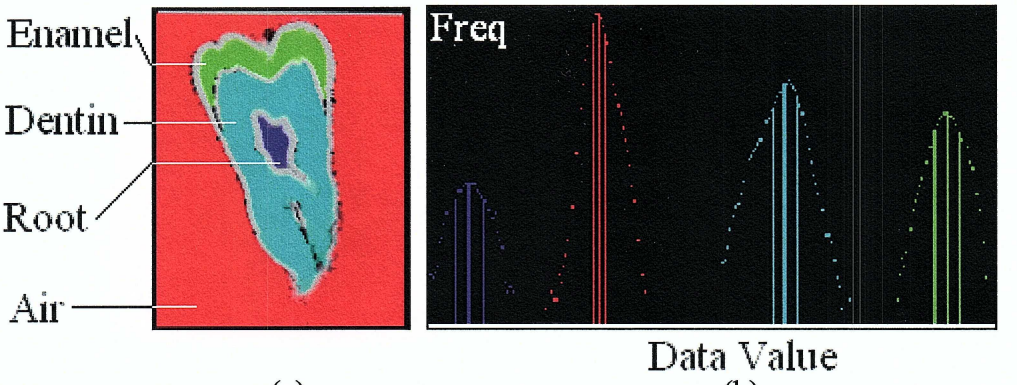


(a) (b) (c)

Figure 27: Sets of adjacent interior voxels in the human tooth data set.

determined a separate material region in the tooth. This is shown by analyzing the frequency distributions of data value for all interior voxels. Figure 27 (b) shows the frequency distribution for all interior voxels within the volume. As expected, there are four distinct regions of high frequency data value which correspond to the four data value ranges of each material. The plot in Figure 27 (c) shows the joint data value frequency distributions for each set of adjacent interior voxels found during the initial step of segmentation, where each adjacent set's data value distribution is graphed in the color assigned that set in Figure 27 (a), and the mean and one standard deviation are shown by vertical bars. Figure 27 (c) also shows a case where two material regions (those in green and purple) are separated in the earliest stage of the segmentation algorithm, but actually correspond to the same material (dentin) in the real-world object and thus should be merged during or after region growing.

After the initial material regions have been determined, the next step is to grow each region and merge regions believed to correspond to the same material in the real-world object. Using the adjacency merge parameter of $J_A = 2.0$, the algorithm grows each material region to include all adjacent voxels within two standard deviations of the material's mean data value. Additionally, any material regions which become adjacent as a result of growing are merged if their standard deviation ranges overlap. Figure 28 (a) shows the result of region growing in the tooth data set. Notice in the joint data value frequency plot of Figure 28 (b) that two of the material regions have become merged during the region growing process. The material shown in light blue in Figure 28 (b) is the result of merging the materials shown in green and purple in Figure 27 (c).



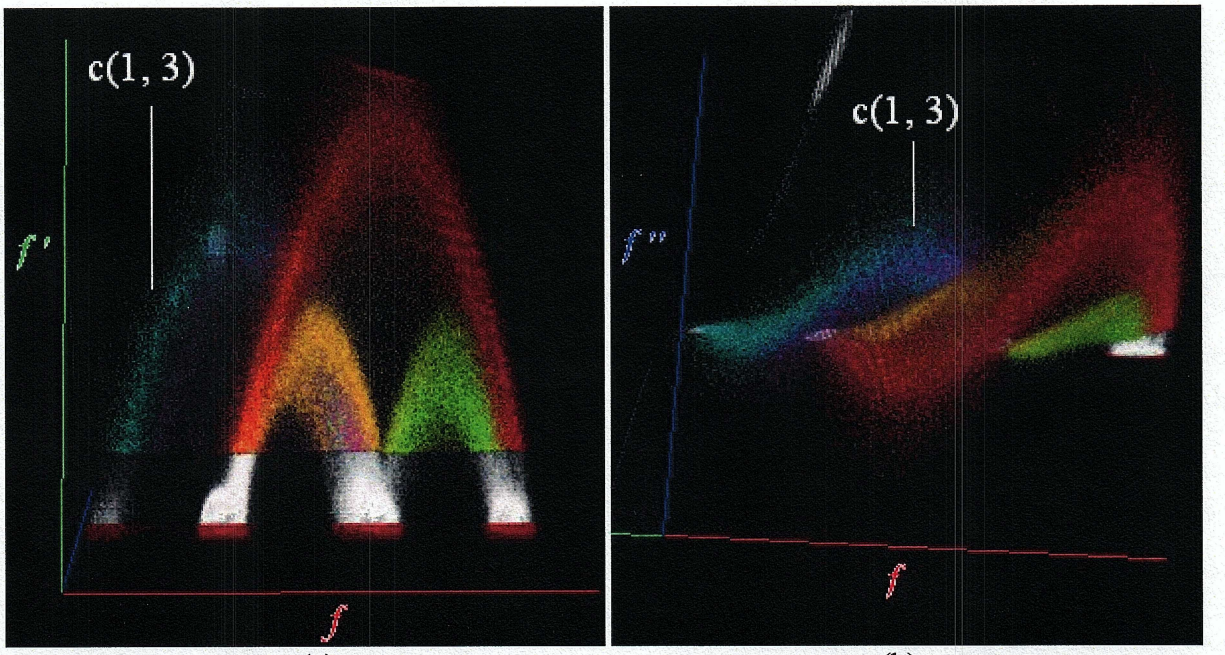
(a) (b)

Figure 28: Material regions after region growing in the human tooth data set.

The next stage of the algorithm determines boundary indices and a position function for each boundary. Boundary indices are assigned to each boundary voxel based on the two materials found in the direction of the gradient and negative gradient vector. For each boundary index (unique combination of two materials) a unique color is chosen from the HSV color space, and a boundary-specific histogram volume is created to be used in creating a position function for that boundary. Figure 29 shows the combination of all boundary-specific histogram volumes generated for the tooth data set, where each bin with f' greater than GM_{bound} is colored according to the boundary it corresponds to. Note that each boundary curve above the GM_{bound} plane in Figure 26 (with the exception of one) is successfully separated in Figure 29. This indicates that each boundary-specific histogram volume generated for each boundary does in fact represent only those bins that correspond to a particular material boundary, and thus minimizes the risk of poor f' and f'' approximations due to overlap. The exception to this, mentioned above, occurs within the boundary between the root and dentin, curve c(1, 3). Figure 29 shows boundary curve c(1, 3) as consisting of two colors, blue and purple, which indicates that the algorithm has determined two distinct boundaries in the volume that actually are the same. The cause for this erroneous distinction is discussed shortly.

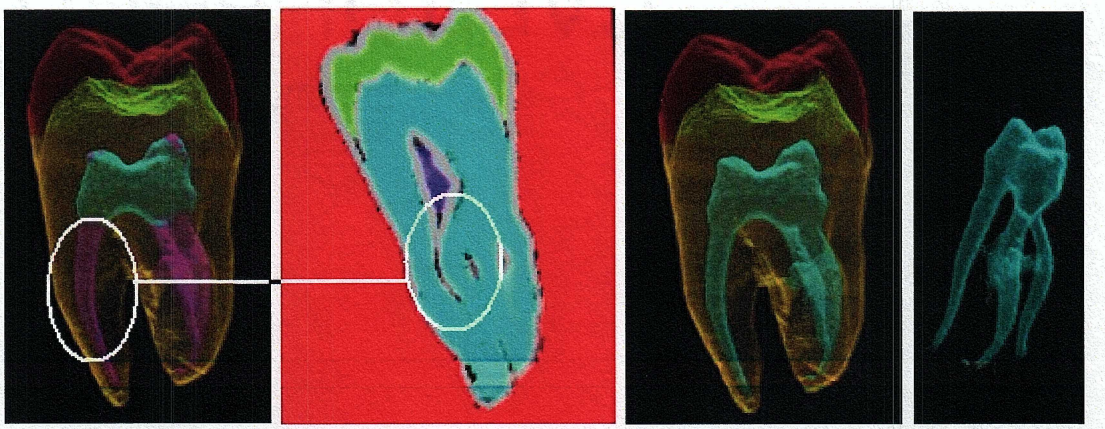
Once boundary-specific histogram volumes have been created, the position function $p(v, b)$ is generated by analyzing the location of the bin centroid at data value v in the histogram volume for boundary index b. For each boundary index b, an opacity function

Opacity^b(x) is automatically generated which maps each position along boundary b to an opacity. The opacity functions, position function, and unique boundary colors then are used to generate the *Optical*(v, b) function which determines the color and opacity for each boundary voxel based on the data value v and boundary index b that voxel is assigned. After Phong shading is applied the final rendering is displayed to the user, as seen in Figure 30 (a).



(a) (b)

Figure 29: Joint histogram volumes of each material boundary in the human tooth.



(a) (b) (c) (d)

Figure 30: Final rendered results of the human tooth data set.

Figure 30 (a) shows that the method proposed is capable of semi-automatically determining a transfer function that clearly shows all material boundaries within the tooth such that opacity is assigned appropriately to boundary voxels and individual material boundaries may be visually distinguished by color. The exception to this is seen in the coloration of the boundary between dentin and root corresponding to boundary curve c(1, 3) circled in Figure 30 (a). The algorithm has found two distinct boundaries (purple and blue) where only one boundary should exist. To understand the cause of this, Figure 30 (b) shows a cross-section of the tooth after region growing. The areas in Figure 30 (b) colored black represent voxels that did not get assigned to a material region as a result of region growing, areas in white represent boundary voxels, and areas in red, yellow, blue, and purple represent each material region. Note that the root material (colored purple in Figure 30 (b)) did fully grow into the region circled in white. As a result, when determining the materials which compose the boundary within the circled region, the algorithm follows the positive and negative gradient vectors for each voxel in the circled region and determines that the boundary circled is between dentin and dentin, rather than between dentin and root. That is, since the root region did not grow completely, some of the boundary voxels between root and dentin were determined to be between dentin and dentin, resulting in a new boundary (shown in purple in Figure 30 (a)). Fortunately, false boundaries such as this can be easily corrected by the user using an interface to manually merge boundaries. Manually merging the two regions results in the rendering shown in Figure 30 (c). Moreover, as a result of separate opacity functions for each boundary, the user can make all other boundaries transparent and view only the boundary between dentin and root, as shown in Figure 30 (d).

5.2 Engine Block

Another volume data set commonly used in demonstration of DVR methods is the CT scan of an engine block. The engine block data set used in this research is a (256, 256, 128) dimensioned volume with one 8-bit scalar value per voxel. Linear bin scaling is performed using the default parameters, yielding the global histogram volume visualized in Figure 31 (a). The histogram volume indicates that there are three materials in the volume, and three boundaries between these materials. This data set proves challenging for the algorithm as it

belies the assumption that each material will have a range of data values that correspond to it. To see this, note that Material 3 in Figure 31 (a) has very small data value range (likely due to aggressive scaling during the data acquisition process) and does not form the characteristic blob of low f' bins in the histogram volume. Despite this, boundary curves c(1,3) and c(2,3) obviously span from Materials 1 and 2 to Material 3, which assures that Material 3 does exist.

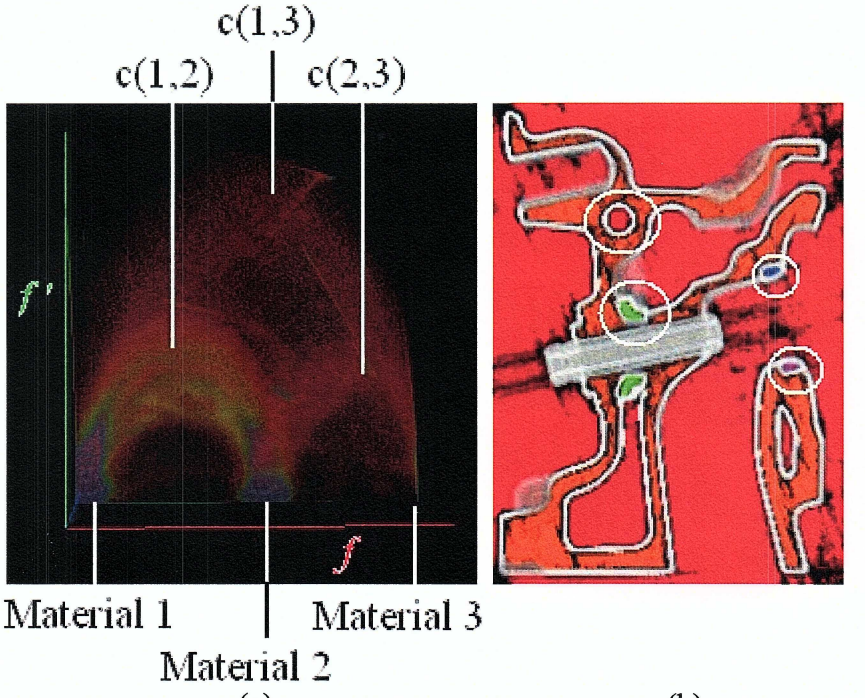
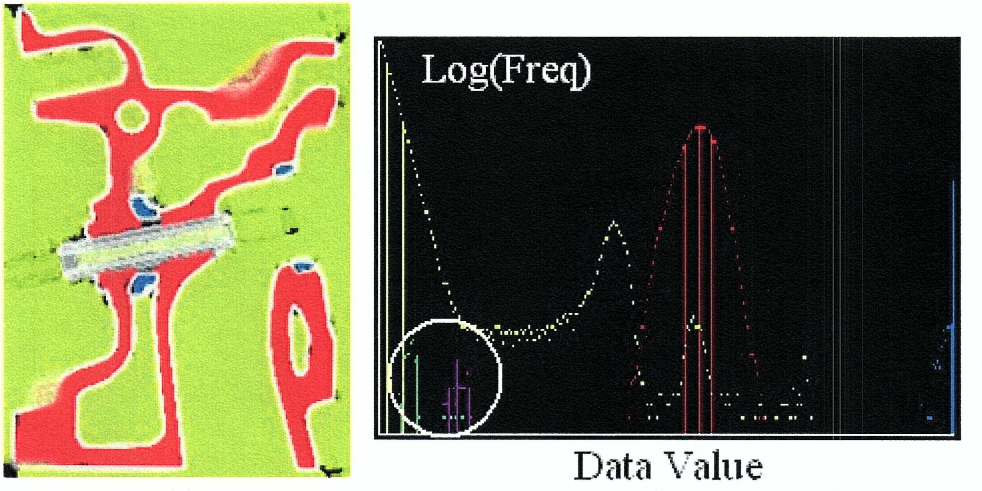


Figure 31: Histogram volume and sets of adjacent interior voxels for engine block data set.

Using gradient magnitude threshold $GM_{mat} = 7$ produces 19 sets of adjacent interior voxels. Each set is shown in a different color in Figure 31 (b), and some of the smaller regions are circled in white for clarity. Regions are then grown and merged using $J_A = 2.0$ and $J_G = 0.5$ resulting in five material regions. Figure 31 (a) shows a cross-section of the engine block as this stage of the algorithm, such that each color represents a material region. Despite finding two more material regions in the volume than are indicated in the histogram volume, material classification is approximately correct as the two "extra" material regions found make up a very small percentage of the total number of material voxels classified. This can be seen in the joint log(frequency) plots of data value for each of the material regions. Figure 32 (b) displays each frequency plot in the color of the corresponding material

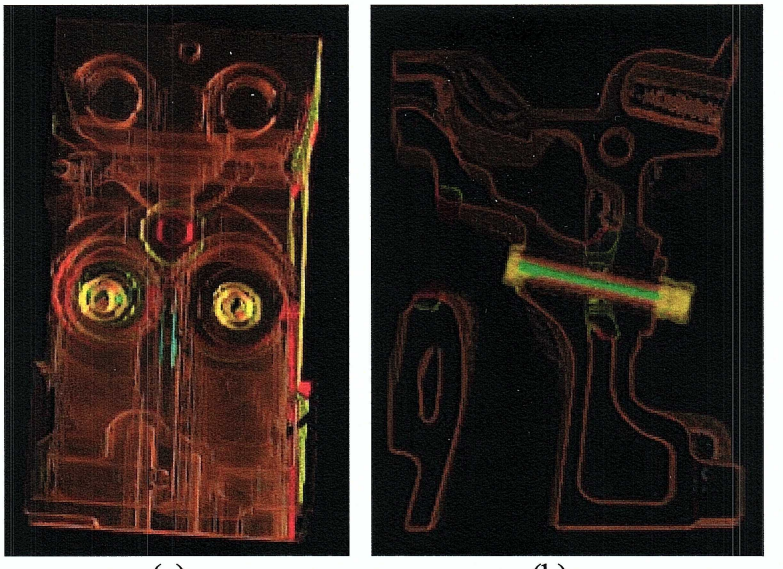
region's color in Figure 32 (a). Note the white circle in the plot that indicates the frequency of the two "extra" regions. Noting that frequency is represented on a logarithmic scale, it is evident that these two materials constitute only a very small percentage of all material voxels, and thus will contribute very little to the final rendering.



(a) (b)

Figure 32: Material regions after region growing in the engine block data set.

Using the gradient vector for each boundary voxel, the algorithm then finds 14 distinct material boundaries, 4 of which are assigned to a significant number of boundary voxels. Figure 33 (a) shows the final rendered image using automatically generated opacity functions and color. Note that the yellow areas near the center of the rendering appear pixelated as a result of the automatically generated opacity function. This is likely due to an extreme minimum or maximum position for that material, which in turn leads to an automatically generated opacity functions whose ramp begins at an unnecessarily extreme position (see Equation 13). This can easily be corrected by the user with an interface that allows for manual modification of opacity functions per boundary. Figure 33 (b) shows a cross-section of the final rendering that clearly visualizes the location and difference of boundaries in the data set.



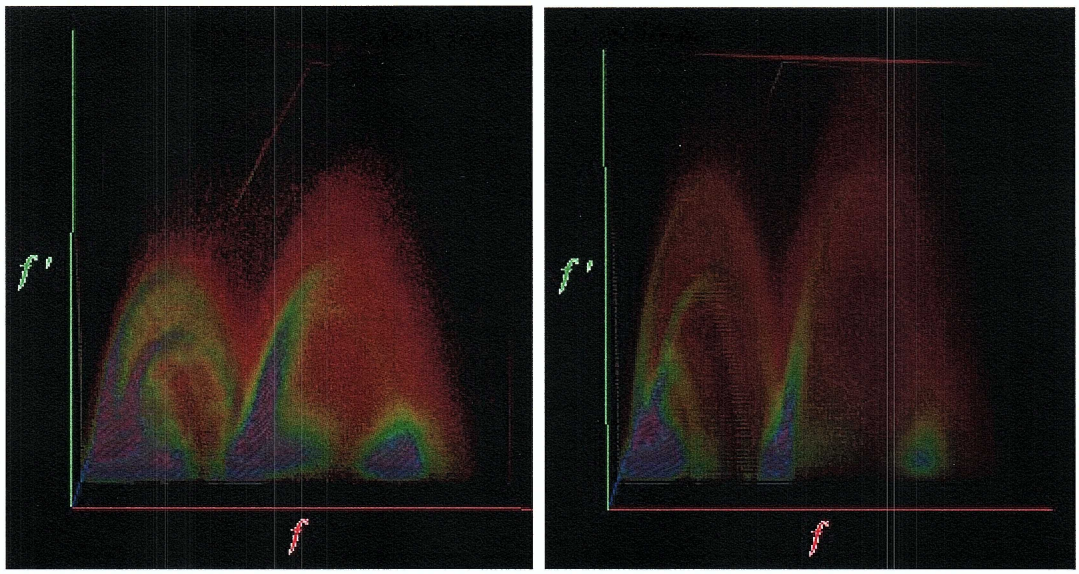
(a) (b)

Figure 33: Final rendered results of engine block data set.

5.3 Human Head

A CT scan of a human head was used as a test of the proposed method's ability to visually distinguish different material boundaries in a more complex data set. The human head CT is a (128, 256, 256) dimensioned volume with one 8-bit scalar data value per voxel and is a subset of the Chapel Hill CT data set.

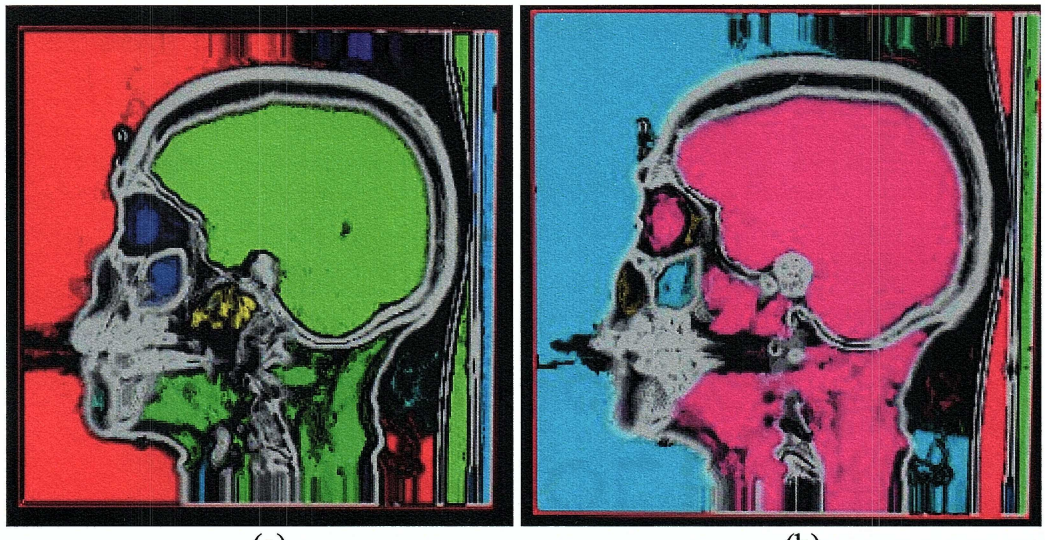
As with the previous data sets examined, the default parameters used in linear histogram bin scaling provide a good fit to the parametric data obtained from the data set, as seen in Figure 34 (a). Refinement of the histogram visualization parameters allows for a more informative understanding of the structure of the histogram volume, shown in Figure 34 (b). Note in Figure 34 (a) that there are multiple boundary curves in the histogram volume, some of which do not appear to start or end at the low f' blobs normally associated with materials. This indicates that there are multiple material boundaries in the volume data, and that some of these boundaries are between materials with relatively low frequency.



(a) (b)

Figure 34: Histogram volume of the human head data set.

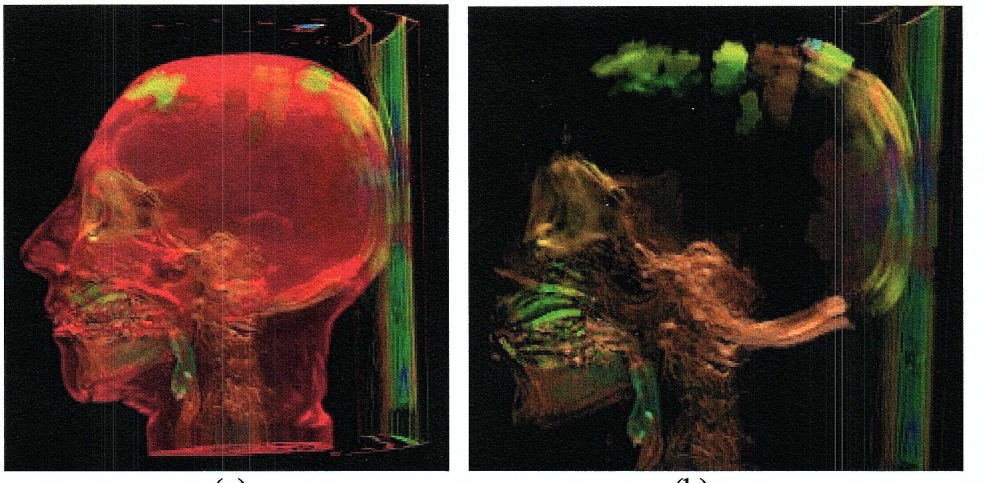
Using parameter settings $GM_{mat} = 5$, and $GM_{bound} = 50$ yields 186 sets of adjacent material voxels. Figure 36 A shows a cross-section of the human head with each set of interior voxels colored differently. Region growing (and merging) is then implemented with merge parameters $J_A = 1.0$ and $J_G = 0.5$, resulting in the merging of 16 regions that became adjacent during region growing and 140 non-adjacent regions, yielding 30 regions total after merging (shown in Figure 35 (b)). Figure 35 (a) shows that sets of adjacent interior voxels do well at delineating various material regions in the human head. Figure 35 (b), however, shows that this delineation is somewhat negated by the merging procedure within region growing. That is, Figure 35 (b) shows that overly aggressive merging has erroneously merged some regions. For example, note that the area near the eye (blue) in Figure 35 (a) is regarded as a separate material from other soft tissues such as the brain (green), however these two material regions are merged during region growing, as evidenced by the fact that they are both rendered in purple in Figure 35 (b). Further, note that after region growing in Figure 35 (b) the area where the skull should be is rendered black, indicating that no region grew into this area. This causes complications for further stages of the algorithm that are addressed below.



(a) (b)

Figure 35: Material regions before and after region growing in the human head data set.

Of the 30 material regions 86 boundary indices are assigned to boundary voxels, opacity functions are created, and the final rendered image is displayed. Figure 36 (a) shows the final rendered image, while Figure 36 (b) shows a cross-section of the head with boundary between air and skin rendered transparent. The final rendering using the parameters mentioned above demonstrates that the position and opacity functions work well in assigning an appropriate opacity to each boundary such that each boundary is visible and underlying boundaries are not occluded. The separation of boundary by color, however, does not perform well on this data set using the given parameters. Particularly noticeable are the multiple colors assigned to the boundary between air and the skin near the top of the head. Recall that Figure 35 (b) indicated that the region representing the skull had not been properly grown during region growing. As a result, when the gradient and negative gradient vectors for voxels in this boundary are followed the materials detected are inconsistent, resulting in the erroneous assignment of various boundary indices to these voxels within the same boundary.



(a) (b)

Figure 36: Final rendered results of human head data set.

Unlike the previous two data sets, the human head CT contains a large number of materials. As mentioned previously, a great deal of research has been devoted to the segmentation and visualization of biomedical volume data (such as this data set) due to the complexity of biological objects and the difficulty of classifying each of the many materials found in them. As such, segmentation methods designed specifically for biomedical data are unquestionably superior to the method presented in this thesis for volume data within that domain. However, the segmentation method proposed here is not intended as a domain-specific means of material classification. Rather, the method proposed favors simplicity over classification accuracy in an attempt to minimize reliance on domain-specific knowledge.

Chapter 6. Conclusion and Future Work

6.1 Conclusion

The goal of this thesis is to provide a non-domain specific, semi-automatic means of generating transfer functions that provide direct volume renderings which informatively display the boundaries between materials. To this end, the method proposed attempts to determine the location of each boundary and render each boundary a different color such that all boundaries are rendered with minimal occlusion. This is done through a non-domain specific segmentation technique based on confidence connected region growing, and an extension of the semi-automatic opacity function generation method proposed by Kindlmann.

Results on the tooth and engine block data set demonstrate that the method proposed works well in volume data sets with relatively few materials and distinct boundaries between materials. Additionally, these results were obtained with a minimum amount of parameter setting and without domain-specific knowledge on the part of the user or the algorithm. Results on the human head CT scan showed the ability of the method to produce informative opacity functions on data sets with a large number of materials and complex material boundaries, but also showed that the simple segmentation scheme presented may not be adequate for such complex volumes.

In summary, the contributions of this thesis can be stated as follows:

1. The method proposed provides a simple, non-domain specific means of rendering material boundaries in an informative fashion.
2. The segmentation method proposed offers a means of classifying materials within a volume that requires only two scalar parameters and does not require domain-specific knowledge or manual seed specification.
3. Linear histogram bin scaling provides a means of determining histogram bin scaling constants such that any volume's parametric data (value, f', and f'') can be fit to a histogram volume in a way that optimizes the information content of the histogram.
4. A histogram volume visualization tool is presented that allows the user to gain a more detailed understanding of the structure of boundary curves in the histogram volume.

5. The ability to produce a histogram volume for each boundary in the volume, as a consequence of segmentation, allows for more accurate approximations of f' and f'' per data value, leading to more accurate opacity function generation.
6. An automatic means of generating opacity functions which map position *per boundary* to opacity is proposed based on Kindlmann's notion that voxels near the center of a material boundary should be rendered more opaque.
7. The ability to map voxels to a boundary index by use of the gradient vector allows for the final rendering to display different colors for each boundary, yielding a more informative rendering. An additional benefit of this mapping is the ability to easily change the optical properties of individual boundaries manually during data exploration.

6.2 Future Work

One of the factors that limited research was the amount of processing time required to run the algorithm presented. Due to the simplicity of its design, the segmentation portion of the algorithm runs relatively quickly even on large data sets. The bottleneck of the method occurs when multiple histogram volumes must be computed. While the run time of the algorithm is linear in terms of the number boundaries found, actual calculation of the histogram volumes is processor intensive and can take upwards of 20 seconds per boundary on a current workstation. Additionally, not all of the boundaries found are significant (due to incomplete region growing), yet the algorithm requires the same amount of time to compute the histogram volume of non-significant boundaries as it does for significant boundaries. Development of a means for determining the significance of a boundary, or simply optimizing the creation of histogram volumes, would lead to decreased run time.

The segmentation portion of the algorithm proved effective on volume data sets with a relatively small number of materials, yet had difficulty with a large number of materials. This is due, in part, to erroneous merging of materials both when they become adjacent during region growing (based on parameter J_A) , and when all materials are merged based on J_G . The improper merging of material regions that become adjacent during region growing can be reduced by decreasing the parameter J_A , however this can cause incomplete region

growth, which in turn causes boundary voxels belonging to the same boundary to be assigned to different boundary indices. This problem of properly merging material regions is compounded on data sets with a large number of materials (such as the human head CT), making this an area in need of future research.

An additional difficulty in research was the amount and type of volume data sets available for experimentation. Although there are a great number of biomedical volume data sets publicly available, the goal of this thesis was to provide a *non-domain specific* method for semi-automatic volume rendering. Without a larger collection of data sets from varying domains it is difficult to determine the effectiveness of the method in a non-domain specific environment. Further, the data sets used in this research all consist of one 8-bit scalar data value per voxel. The use of 16-bit (or larger) data may increase the accuracy of the segmentation method as more precise data values may help delineate materials during classification.

Finally, the method presented attempts to minimize the number of parameters that must be set in order to obtain an informative rendering. Despite this, the information content of the final rendered image is highly dependent on the few parameters used by the proposed method (particularly, GM_{mat}, J_A, and J_G). Manual alteration of these parameters can be time consuming, and the relation between the values of these parameters and changes in the rendered image is not exceptionally intuitive. The use of supervised artificial intelligence algorithms or unsupervised algorithms in conjunction with user-defined objective functions could aid in the selection of these parameters and further simplify the method presented.

Bibliography

[SMG03] Pedro Mario Silva, Marcos Machado, and Marcelo Gattas. 3D Seismic Volume Rendering. In *Congresso da Sociedade Brasileira de Geofisica*, July 2003.

[Lai95] David H. Laidlaw. *Geometric Model Extraction from Magnetic Resonance Volume Data*. PhD thesis, Department of Computer Science, California Institute of Technology, Pasadena, CA, May 1995.

[LC87] W.E. Lorenson and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *ACM Computer Graphics (SIGGRAPH '87 Proceedings)*, pages 163-169, July 1987.

[Lev88] Marc Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications*, 8(5):29-37, 1988.

[Pho75] Bui-Tong Phong. Illumination for Computer-Generated Pictures. *Communications of the ACM*, 18(6):311-317, June 1975.

[ISNC03] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide: The Insight Segmentation and Registration Toolkit*. Kitware, Inc, 2003.

[LM04] Eric B. Lum and Kwan-Liu Ma. Lighting Transfer Functions Using Gradient Aligned Sampling. In *Proceedings Visualization '04*, pages 289-296, 2004.

[PLSea00] Hanspeter Pfister, Bill Lorensen, Will Schroeder, Chandrajit Bajaj, Gordon Kindlmann. The Transfer Function Bake-Off (Panel Session). In *Proceedings Visualization '04*, pages 523-526, 2000.

[MHBea00] Michael Meißner, Jian Huang, Dirk Bartz, Klaus Mueller, Roger Crawfis. A Practical Evaluation of Popular Volume Rendering Algorithms. In *Proceedings 2000 IEEE Symposium on Volume Visualization*, pages 81-90, 2000.

[KKH02] Joe Kniss, Gordan Kindlmann, and C. Hansen. Multidimensional Transfer Functions for Interactive Volume Rendering. *IEEE TVCG*, 8(3):270-285, 2002.

[HHKea96] Taosong He, Lichan Hong, Arie Kaufman, and Hanspeter Pfister. Generation of Transfer Functions with Stochastic Search Techniques. In *Proceeding Visualization '96*, pages 227-234, 1996.

[KUAea05] Joe Kniss, Robert Van Uitert, Abraham Stephens, Guo-Shi Li, Tolga Tasdizen, and Charles Hansen. *Statistically Quantitative Volume Visualization*. In *Proceedings IEEE Visualization 2005*, 2005.

[CLOea05] J. Cerquides, M. Lopez-Sanchez, S. Ontanon, E. Puertas, A. Puig, O. Pujol, and D. Tost. Learning Method for Automatic Classification of Biomedical Volume Datasets. In *Conferencia de la Asociacion Espanola para la Inteligencia Artificial*, 2005.

[Kin99] Gordon Kindlmann. *Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering*. Master's thesis, Program of Computer Graphics, Cornell University, Ithaca, NY, 1999.

Acknowledgements

I would firstly like to thank God for the blessings given me and the opportunities afforded me. Special thanks to my mother, father, brothers, and family who have encouraged and supported me throughout my long journey in academia. Thanks to Dirk Reiners and Chris Harding for their enthusiasm, technical know-how, and patience as I overcame the learning curve of computer graphics. And finally, a thank you to the Crowder family and the Jensen family for their support, generosity, and kindness.